

# Supplementary Material for GigaHands: A Massive Annotated Dataset of Bimanual Hand Activities

The supplementary material for GigaHands provides additional details and results to support the main paper. It includes supplementary videos, examples of data and annotations, comprehensive comparisons of dataset statistics, detailed explanations of our hand and object motion tracking methods, elaborations on text instructions and annotations (including prompts and examples), further experiments on text-driven motion synthesis, motion captioning, and dynamic reconstruction. We also provide additional applications on hand-object motion tasks, as well as detailed inspections of the dataset, such as object visualizations, the verb pool, and lists of scenarios and scenes.

## Contents

<b>1. Data and Annotation Example</b>	<b>2</b>
<b>2. Comparison of Dataset Statistics</b>	<b>3</b>
<b>3. Experiments on Text-driven Motion Synthesis</b>	<b>3</b>
<b>4. Experiments on Motion Captioning</b>	<b>6</b>
<b>5. Experiments on Dynamic Reconstruction</b>	<b>8</b>
<b>6. Experiments on Hand-Object Interaction</b>	<b>9</b>
<b>7. Hand Motion Tracking</b>	<b>10</b>
<b>8. Object Motion Tracking</b>	<b>11</b>
<b>9. Text Instructions and Annotations</b>	<b>13</b>
9.1. Prompts and Examples for scenario grouping . . . . .	13
9.2. Prompts and Examples for Scene structuring . . . . .	13
9.3. Prompts and Examples for instruction scripting . . . . .	15
9.4. Annotation Interface . . . . .	15
9.5. Prompts and Examples for text augmentation . . . . .	15
<b>10Dataset Inspection</b>	<b>16</b>
10.1Object Visualization. . . . .	16
10.2Verb Pool. . . . .	17
10.3List of Scenarios and Scenes. . . . .	18

# 1. Data and Annotation Example

Table 1 illustrates the types of annotations we provided for a single motion clip. These include the text instruction, text annotation, augmented annotations, original multi-view RGB videos, 2D and 3D keypoints, 3D hand meshes, hand motions, object masks, textured object meshes, and object motions.


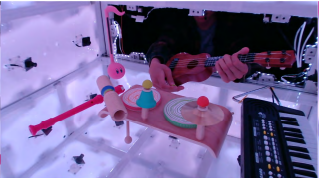


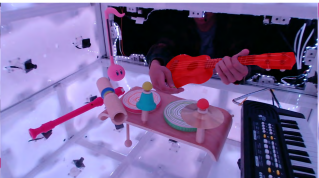


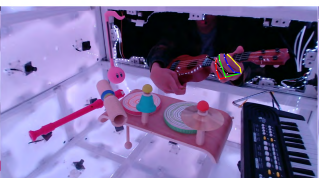

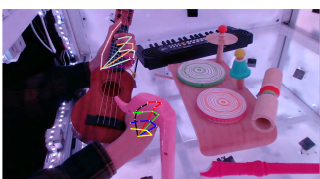
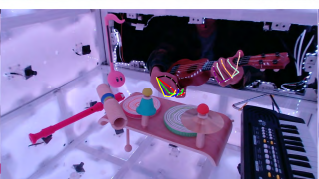
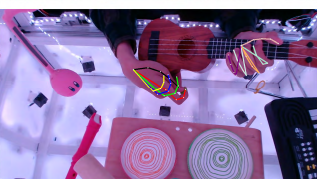
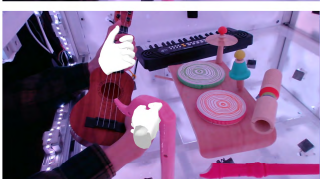

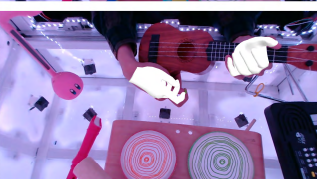


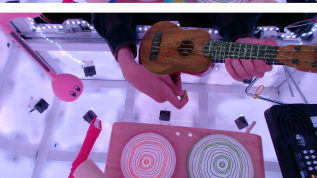
type	view #1	view #2	view #3
instruction	Place your left-hand fingers on the fretboard of the ukelele to form chords.		
annotation	Place your left-hand fingers on the fretboard of the ukelele to form chords.		
augmented annotation	Position your left-hand fingers on the fretboard to shape chords.		
	Rest your left-hand fingers on the fretboard to create chords.		
	Set your left-hand fingers on the fretboard to assemble chords.		
	Arrange your left-hand fingers on the fretboard to build chords.		
	Use your left-hand fingers on the fretboard to construct chords.		
video			
object masks			
2D hand keypoints			
3D hand keypoints			
3D hand mesh/motion			
3D object mesh/motion			

Table 1. Annotations for a single motion clip.

## 2. Comparison of Dataset Statistics

Table 2. Comparisons of 3D bimanual motion datasets. Dataset names are highlighted with different colors if it has no text annotations (gray), **action type** (green), **sparse description** (red), and **dense description** (blue).

Name	setting	markerless	hand track.	object track.	#mins	#motions	# poses	#views	#frames	#subjects	#objects	#verbs
<b>AssemblyHands</b> [22]	studio	✓	semi-auto.	/	630	62	203k	12	3.03M	34	/	24
<b>Ego4D</b> [13]	in-the-wild	✓	/	/	/	/	/	1	/	931	/	1452
<b>Ego-Exo4D</b> [7]	in-the-wild	✓	manual	/	/	/	<b>4.4M</b>	5-6	/	<b>740</b>	/	402
<b>HOI4D</b> [17]	in-the-wild	✓	manual	manual	1,333	4k	1.2M	1	2.4M	4	<b>800</b>	25
ARCTIC [5]	studio	✗	mocap	mocap	121	339	218k	9	2.1M	10	11	✗
TACO [18]	studio	✗	mocap	mocap	202	2.3k	363k	13	4.7M	14	196	13
<b>OakInk2</b> [39]	studio	✗	mocap	mocap	557	2.8k	993k	4	4.01M	9	75	55
HOT3D [3]	studio	✗	mocap	mocap	833	4.1k	1.7M	2-3	3.7M	19	33	✗
<b>GigaHands (Ours)</b>	studio	✓	auto	auto	<b>2,034</b>	<b>13.9k</b>	3.7M	<b>51</b>	<b>183M</b>	56	417	<b>1467</b>

Table 2 provides a comprehensive comparison of 3D bimanual motion datasets, including their capturing sources, annotation methods, and statistics across various features. The table applies the same method as Figure 2 in the main paper for verb counting. To compile the verb counts, we extracted verbs from several sources: fine-granularity labels from AssemblyHands [22], atomic descriptions from Ego-Exo4D [7], redacted narrations from Ego4D [6], category labels and task definitions from HOI4D [17], action types from TACO [18], task descriptions and affordance annotations from OakInk2, and our own augmented text descriptions from GigaHands. For verb extraction, we parsed sentences using spaCy [11] and collected the verb stems. Verb stems that were misspelled or not recognized as verbs in either WordNet [20] or spaCy’s ‘en\_core\_web\_sm’ model were removed. Clearly, GigaHands surpasses all other datasets in filming length, number of motion sequences, number of camera views, frame count, and verb count.

## 3. Experiments on Text-driven Motion Synthesis

**Implementation Details** Our framework builds upon the T2M-GPT architecture proposed by [40] and utilizes their publicly available PyTorch codebase<sup>1</sup>. The network comprises a motion VQ-VAE that learns a mapping between motion data and discrete code sequences and a T2M-GPT which generates code indices conditioned on the text description. For the Motion VQ-VAE, the codebook size is set to  $512 \times 512$ , with a downsampling rate  $l = 4$ . The encoder and decoder is a simple convolutional architecture consisting of 1D convolutions, residual blocks [10], and ReLU activations. Temporal downsampling and upsampling are performed using strided convolutions (stride = 2) and nearest-neighbor interpolation, respectively. We use the AdamW optimizer [19] with  $[\beta_1, \beta_2] = [0.9, 0.99]$ , a batch size of 256, and an exponential moving constant  $\lambda = 0.99$ . The model is trained for the first 200K iterations with a learning rate of  $2 \times 10^{-4}$ , followed by 100K iterations with a reduced learning rate of  $1 \times 10^{-5}$ . Based on our text annotations, we construct a custom word vectorizer utilizing pre-trained 300-dimensional word embedding vectors from GloVe [28].

For the T2M-GPT, we employ adopts a transformer architecture [36] consisting of 18 layers with a model dimensionality of 1,024 and 16 attention heads. The maximum length of the code index sequence is capped at 50, with an additional end token incorporated to indicate sequence termination. The optimization of the transformer is performed using the AdamW algorithm [19], configured with hyperparameters  $[\beta_1, \beta_2] = [0.5, 0.99]$  and a batch size of 128. The training process spans 150K iterations with an initial learning rate of  $1 \times 10^{-4}$ , followed by a decay to  $5 \times 10^{-6}$  over an additional 150K iterations.

To ensure consistency across three datasets—TACO [18], OakInk2 [39], and GigaHands—we standardize the hand representations and motion ranges. First, we extract 3D hand keypoints from the MANO [31] parameters using the MANO Layer<sup>2</sup>. Subsequently, we align the hand orientations such that the fingertips point in the positive z-axis direction. Additionally, we recenter each motion sequence by adjusting the hand positions so that the motion center of both hands is aligned to the origin.

**Evaluation Metrics** Given the lack of a standard hand motion feature extractor, we train a simple framework consisting of a motion extractor and a text extractor trained under a contrastive learning paradigm, following [8]. The Motion and Text Feature Extractors are designed to learn geometrically close feature vectors for matched text-motion pairs while ensuring separation for mismatched pairs. Specifically, the input text and motion are encoded into semantic vectors  $F_t$  and  $F_m$ ,

<sup>1</sup><https://github.com/Mael-zys/T2M-GPT>

<sup>2</sup><https://github.com/hassony2/manopt>

respectively, using two distinct bi-directional GRUs. To achieve this, we minimize a contrastive loss that enforces proximity for matched pairs and imposes a margin of separation  $m$  for mismatched pairs:

$$L_{cst} = y \cdot (\max(0, m - \|F_t - F_m\|_2^2))^2 + (1 - y) \cdot \|F_t - F_m\|_2^2, \quad (1)$$

where  $y \in 0, 1$ , with  $y = 1$  indicating matched text-motion pairs and  $y = 0$  otherwise. The margin  $m$  is set to 10 across all datasets. These feature extractors are independently trained for each dataset to establish an upper bound respectively.

The trained text and motion feature extractors are utilized to evaluate text-to-motion generation using the metrics proposed in [8]. We denote the ground-truth motion features, generated motion features, and text features derived from the aforementioned feature extractors as  $F_{gt}$ ,  $F_{gen}$ , and  $F_t$ , respectively.

**R Precision.** For each generated motion, a text pool is formed with its ground truth text and 31 randomly selected mismatched texts. Euclidean distances between the description feature and motion features are computed and ranked. If the ground truth text ranks in the top-k positions ( $k=1, 2, 3$ ), it counts as a successful retrieval. The average accuracy over all samples defines the top-k R-precision.

**Multimodal Distance (MM Dist.)** MM Dist. evaluates the alignment between text embeddings and generated motion features. Given  $N$  randomly generated samples, it calculates the average Euclidean distance between each text feature and the corresponding generated motion feature:

$$\text{MM Dist.} = \frac{1}{N} \sum_{i=1}^N \|F_{gen}^i - F_t^i\| \quad (2)$$

where  $F_{gen}^i$  and  $F_t^i$  are the features of the  $i$ -th text-motion pair.

**Frechet Inception Distance (FID).** FID measures the distributional similarity between real and generated motion features. Features are extracted from ground-truth motions in the test set and generated motions from the corresponding descriptions. FID is computed as:

$$\text{FID} = \|\mu_{gt} - \mu_{gen}\|^2 - \text{Tr}(\sigma_{gt} + \sigma_{gen} - 2(\sigma_{gt}\sigma_{gen})^{\frac{1}{2}}), \quad (3)$$

where  $\mu_{gt}$  and  $\mu_{pred}$  are mean of  $F_{gt}$  and  $F_{gen}$ .  $\sigma$  is the covariance matrix and  $\text{Tr}$  denotes the trace of a matrix.

**Diversity.** Diversity quantifies the variance across all generated motion sequences in the dataset. From the generated motions,  $S_{dis}$  pairs of motion features are randomly sampled, denoted as  $F_{gen}^i$  and  $F_{gen}^{i'}$ . The diversity is then calculated as:

$$\text{Diversity} = \frac{1}{S_{dis}} \sum_{i=1}^{S_{dis}} \|F_{gen}^i - F_{gen}^{i'}\|, \quad (4)$$

In our experiments, we set  $S_{dis} = 300$ , as suggested in [8].

**MultiModality.** MultiModality assesses the diversity of hand motions generated from the same text description. For the  $i$ -th text description, 20 motions are generated, and two subsets, each containing 10 motions, are sampled. Denoting the features of the  $j$ -th pair for the  $i$ -th text description as  $(F_{gen}^{i,j}, F_{gen}^{i,j'})$ , MultiModality is computed as:

$$\text{MultiModality} = \frac{1}{10N} \sum_{i=1}^N \sum_{j=1}^{10} \|F_{gen}^{i,j} - F_{gen}^{i,j'}\| \quad (5)$$

**Comparison of Different Backbones.** We evaluate the text-to-hand motion synthesis performance on our dataset using three different backbone models: TM2T [9], MDM [35], and T2M-GPT [40], as summarized in Table 3. TM2T comprises a motion VQ-VAE module for motion quantization and an attentive GRU-based model for text-to-motion generation. MDM employs a classifier-free diffusion generative approach. While these models were originally developed for human motion synthesis, we adapted them to generate hand motions on our dataset.

From Table 3, it is evident that our dataset supports all three backbones effectively. Notably, the state-of-the-art T2M-GPT architecture for human motion generation also achieves the best performance on our dataset, demonstrating its superior capability for motion synthesis.



Table 3. Quantitative results for text-driven motion synthesis with different backbones trained on our dataset. *upper bound* indicates performance calculated with the ground truth. We repeat the evaluation 20 times and report the average with 95% confidence interval.

Dataset	R Precision(%) $\uparrow$			MM Dist. $\downarrow$	FID $\downarrow$	Diversity $\rightarrow$	MultiModality $\uparrow$
	@1	@2	@3				
upper bound	77.4 $\pm$ .002	88.8 $\pm$ .002	91.3 $\pm$ .001	2.96 $\pm$ .005	0.002 $\pm$ .000	11.9 $\pm$ .097	-
MDM [35]	22.5 $\pm$ .004	42.7 $\pm$ .005	50.2 $\pm$ .005	7.81 $\pm$ .082	5.60 $\pm$ .126	9.8 $\pm$ .088	8.52 $\pm$ .103
TM2T [9]	24.1 $\pm$ .002	38.4 $\pm$ .003	47.1 $\pm$ .004	9.28 $\pm$ .017	8.60 $\pm$ .161	9.7 $\pm$ .065	6.29 $\pm$ .085
T2M-GPT [40]	<b>31.2</b> $\pm$ .003	<b>44.7</b> $\pm$ .004	<b>53.1</b> $\pm$ .004	<b>6.68</b> $\pm$ .023	<b>4.70</b> $\pm$ .078	<b>10.5</b> $\pm$ .090	<b>9.11</b> $\pm$ .255

Table 4. Ablation study on different hand motion representations and text annotations for text-driven motion synthesis. *KP* refers to the 3D hand keypoints representation, while *6D* denotes the MANO pose parameters encoded in the 6D representation. Numbers in parentheses indicate the quantity of text scripts. *upper bound* indicates performance calculated with the ground truth. We repeat the evaluation 20 times and report the average with 95% confidence interval.

Dataset	R Precision(%) $\uparrow$			MM Dist. $\downarrow$	FID $\downarrow$	Diversity $\rightarrow$	MultiModality $\uparrow$
	@1	@2	@3				
upper bound	73.3 $\pm$ .003	87.0 $\pm$ .002	92.1 $\pm$ .001	3.29 $\pm$ .006	0.002 $\pm$ .000	12.8 $\pm$ .067	-
6D (14k)	26.5 $\pm$ .003	43.3 $\pm$ .004	52.3 $\pm$ .003	7.56 $\pm$ .051	5.11 $\pm$ .183	9.7 $\pm$ .078	7.28 $\pm$ .059
6D (84k)	29.5 $\pm$ .003	<b>46.9</b> $\pm$ .004	<b>57.3</b> $\pm$ .004	7.04 $\pm$ .041	<b>4.34</b> $\pm$ .221	<b>11.7</b> $\pm$ .068	9.08 $\pm$ .060
upper bound	77.4 $\pm$ .002	88.8 $\pm$ .002	91.3 $\pm$ .001	2.96 $\pm$ .005	0.002 $\pm$ .000	11.9 $\pm$ .097	-
KP (14k)	27.2 $\pm$ .003	40.1 $\pm$ .002	49.7 $\pm$ .004	7.13 $\pm$ .020	5.20 $\pm$ .169	8.7 $\pm$ .067	7.29 $\pm$ .085
KP (84k)	<b>31.2</b> $\pm$ .003	44.7 $\pm$ .004	53.1 $\pm$ .004	<b>6.68</b> $\pm$ .023	4.70 $\pm$ .078	10.5 $\pm$ .090	<b>9.11</b> $\pm$ .255

**Ablations of Different Motion Representations and Text Annotations.** We further explore the performance of text-to-motion generation using different hand motion representations in Table 4. One approach leverages the 3D hand keypoints derived from MANO parameters, represented as  $X \in \mathbb{R}^{42 \times 3}$ . The other employs the MANO hand pose parameters  $\theta \in \mathbb{R}^{16 \times 6}$ , encoded in a 6D representation [42]. We also conduct an ablation study to evaluate the impact of the number of text descriptions on text-to-motion generation. In GigaHands, each motion clip is paired with six distinct text descriptions, resulting in a total of 84k annotations. We compare the generation results using 14k annotations versus the full set of 84k annotations, as presented in Table 4.

The results demonstrate that both the 6D representation and the 3D keypoint representation achieve comparable performance in text-driven hand motion synthesis. This finding suggests that both representations are equally capable of capturing the essential hand motion features required for this task. Nevertheless, incorporating additional text annotations significantly boosts performance across all evaluation metrics. This improvement underscores the importance of enriched textual descriptions in enhancing the semantic alignment between textual inputs and generated hand motions, irrespective of the chosen motion representation.

**More Qualitative Results.** We present additional qualitative results from our testing sets to demonstrate the effectiveness of text-driven hand motion synthesis in Figure 1. These examples further highlight the ability of our approach to generate semantically aligned hand motions from diverse textual inputs.

test\train	(FID / Diversity)		
	GigaHands	Oakink2	TACO
GigaHands	6.61 / 10.5	44.8 / 4.01	52.1 / 7.11
Oakink2	19.1 / 6.45	19.6 / 6.88	58.1 / 5.04
TACO	22.5 / 11.5	33.6 / 9.31	11.0 / 11.1

Table 5. Cross-dataset evaluation of text-to-motion models trained on different datasets. Each column represents the training dataset, while each row shows the evaluation results on a different dataset. Metrics include FID and Diversity, computed on the test set feature extractor. The ground truth Diversity metrics for GigaHands, Oakink, and TACO are 11.9, 9.30, and 14.2, respectively.



Figure 1. More qualitative results for text-driven motion synthesis on GigaHands. Darker color indicates later frame in the sequence.

**Cross Validation on different datasets.** To validate the capacity of text-to-motion generation models trained on GigaHands, we conduct cross-dataset validation. Specifically, we train T2M-GPT models on one dataset and evaluate their performance on others. Table 5 presents the quantitative results, where each column indicates the dataset used for training, and each row corresponds to the dataset on which the model is tested. We report the FID and Diversity metrics, computed on the corresponding test dataset’s feature extractor.

Since the triplet-based textual annotations in the TACO dataset are coarse, with each triplet corresponding to multiple motion sequences, we employ ChatGPT-4o [2] to refine the captions during testing and select the best-performing generation for each triplet. The results in Table 5 demonstrate that our models achieve competitive performance on unseen datasets, despite being trained on a single dataset. However, due to the simplistic nature of TACO’s annotations, models trained on other datasets struggle to perform well when tested on TACO.

For OakInk2, object descriptions rely on material and texture distinctions rather than explicit shape descriptions. For example, “yellow striped bottle” refers to a large plastic jar, whereas “white bottle” describes a small bottle. This captioning style differs from the textual descriptions in our dataset, leading to a slight performance drop when models trained on our dataset are tested on OakInk2.

## 4. Experiments on Motion Captioning

**Implementation Details.** Our motion captioning framework is also built upon the TM2T [9] architecture, leveraging the same VQ-VAE for motion quantization as used in the text-driven motion synthesis task. For tokenized motion representation, we employ a transformer model to efficiently map hands to textual descriptions. The transformers have 4 attention layers, both with 8 attention heads with 512 hidden size.

When evaluate captioning performance across datasets, we train the VQ-VAE and motion-to-text models separately on TACO [18], OakInk2 [39], and GigaHands. For testing the model’s ability in in-the-wild motion captioning, we first train a VQ-VAE using all datasets combined, followed by training the motion-to-text model exclusively on GigaHands. When generating captions for other datasets, tokenized motion sequences from the combined VQ-VAE are processed by our motion-to-text model, enabling consistent inference across diverse motion data.

**Evaluation Metrics.** We use **R Precision** and **Multimodal Distance (MM Dist.)** to quantitatively measure the performance of our motion-to-text mapping. Unlike in text-to-motion tasks, where motion features are used to retrieve text, we reverse the process by using text features to retrieve the corresponding motion. This adaptation ensures the metrics effectively measure the alignment in motion-to-text tasks. For linguistic evaluation, we use the NLPEval codebase<sup>3</sup> to compute

<sup>3</sup><https://github.com/Maluuba/nlg-eval>

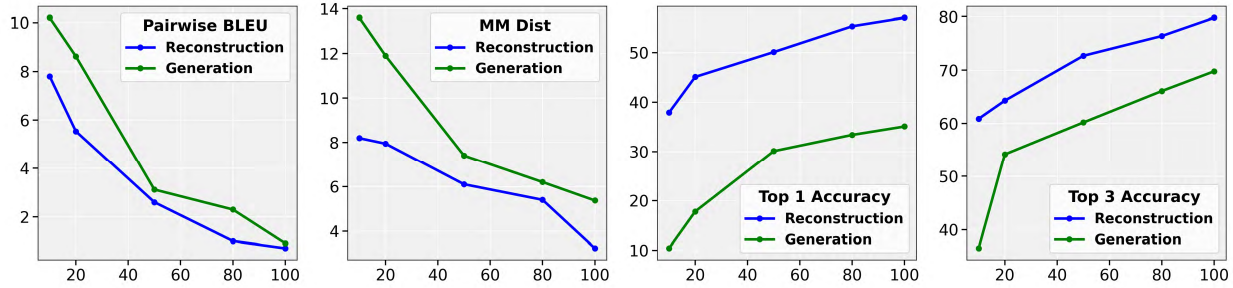


Figure 2. Effect of dataset size on motion reconstruction and motion-to-text generation performance. The x-axis shows the percentage of training data used (10%, 20%, 50%, 80%, and 100%), and the y-axis displays performance metrics: Pairwise BLEU, MM Dist., Top-1, and Top-3 accuracy. Larger datasets consistently improve performance across all metrics, highlighting the benefits of increased data scale.

BLEU [26] and ROUGE [15]. To assess text diversity, we compute Distinct-n[14], which evaluates diversity by counting the number of distinct unigrams and bigrams in the generated texts. Additionally, we measure Pairwise BLEU[34] using the SacreBLEU<sup>4</sup>.

**Impact of Dataset Size.** We show the influence of dataset size on both motion reconstruction and text-to-motion tasks in the paper. Figure 2 further illustrates the impact of dataset size on motion reconstruction and motion-to-text generation. Based on the TM2T architecture, we train a motion VQ-VAE for reconstruction and a transformer-based motion-to-text model for captioning with varying proportions of the training set (10%, 20%, 50%, 80%, and 100%), while evaluating performance on the same test set. We report Pairwise BLEU, MM Dist., and Top-1/Top-3 accuracies.

The results show consistent improvements across all tasks as the dataset size increases. Larger datasets lead to better motion-text alignment (lower MM Dist.), more diverse text generation, and improved retrieval accuracies. These findings emphasize the importance of large-scale data for enhancing performance in motion reconstruction, text-to-motion, and motion-to-text generation.









Table 6. Ablation study on different hand motion representations and text annotations for motion captioning task. *KP* refers to the 3D hand keypoints representation, while *6D* denotes the MANO pose parameters encoded in the 6D representation. Numbers in parentheses indicate the quantity of text scripts. Upper bound indicates the metric performance calculated with the ground truth.

Datasets	R Precision(%) $\uparrow$			MM Dist. $\downarrow$	Pairwise BLEU $\downarrow$	B@4 $\uparrow$	ROUGE $\uparrow$	distinct-1(%) $\uparrow$	distinct-2 $\uparrow$ (%)	BScore $\uparrow$
	@1	@2	@3							
upper bound	75.7	87.2	92.1	3.28	-	-	-	-	-	-
6D (14k)	49.8	62.8	70.2	5.66	1.30	33.7	51.2	4.03	16.4	47.1
6D (84k)	50.2	61.3	66.7	6.31	<b>0.804</b>	36.0	53.7	7.27	24.5	50.3
upper bound	75.3	89.1	93.9	2.87	-	-	-	-	-	-
KP (14k)	<b>57.2</b>	<b>68.9</b>	<b>74.4</b>	<b>4.69</b>	1.21	41.3	56.8	6.28	21.4	52.4
KP (84k)	57.0	66.1	69.8	5.37	0.916	<b>43.1</b>	<b>57.7</b>	<b>15.3</b>	<b>36.9</b>	<b>55.4</b>

**Ablations of Different Motion Representations and Text Annotations.** We also conduct ablation study on different motion representation and number of text annotations for motion captioning task in Table 6

The results indicate that the 6D representation slightly underperforms compared to the keypoint representation in the motion captioning task. However, the addition of extra text annotations significantly improves performance on linguistic metrics, boosting the diversity of generated motions. Also as the number of text annotations per motion clip increases, the retrieval accuracy for matching motions to corresponding text slightly decreases. This could be due to the fact that with more annotations, the matching process becomes more challenging, as the model may have to distinguish between a larger variety of possible descriptions for the same motion, leading to more potential mismatches in retrieval.

<sup>4</sup><https://github.com/mjpost/sacrebleu>

motion									
GT	#1	Raise the brightness control bar to lighten the screen	Move the brightness control bar up to make the screen brighter	Grip the color pen to prepare for writing	Clutch the pen to begin writing	Cross your index and middle fingers on both hands	Insert the Type-C power supply into the laptop	Open the lid of the hand sanitizer bottle	
	#2	Swipe up to show the password screen		Seize one earring with your finger	Place your index and middle fingers in a crisscross position on both hands		Connect the Type-C power supply to the laptop	Uncap the moisturizer container	Twist off the cap of the hand sanitizer bottle








motion								
GT	#1	Unscrew the cap of the large black and yellow striped bottle	Unscrew the cap of the large black and white striped bottle	Rearrange the alcohol burner from one position to another	Rearrange the beaker from one position to another	(empty, cup, bowl)	Open, laptop	Grasp, notebook
	#2	Take off the lid of the water bottle		Pick up the salt container		(empty, cup, bowl)	Lift open the laptop	Grip the notebook
						Pour the cream into the teacup containing the steeped tea	Open the lid of the laptop	Clutch the notebook

Figure 3. More qualitative results for motion caption. Each column shows a motion sequence, its ground truth text description, and two generated texts. Hand motions highlighted in **green**, **orange**, **pink** and **blue** come from **OakInk2**, **TACO**, **Arctic** and **GigaHands**, respectively. Texts highlighted in these colors are generated by models trained on the corresponding datasets. The model trained on **GigaHands** generates diverse captions from a single motion (first row) and accurately captions motions from other datasets (second row). Darker color indicates later frame in the sequence.

**More Qualitative Results.** We present additional qualitative results for motion captioning on our dataset and in-the-wild scenarios (Figure 3). Captions in matching colors are generated by models trained on these datasets. Notably, the model on GigaHands generates diverse descriptions from a single motion and accurately captions motions from other datasets, demonstrating its robustness and generalization ability. For the ARCTIC dataset [5], we apply a similar processing procedure to align orientation and motion range, ensuring consistent 3D hand keypoints. Additionally, we leverage verb annotations from [4] to obtain semantically meaningful motion clips for training the combined VQ-VAE model.

## 5. Experiments on Dynamic Reconstruction

**More Qualitative Results.** Figure 4 presents additional qualitative results on novel view synthesis using GigaHands data with 2D Gaussian Splatting [12]. Each example shows two test views at three different time steps. By utilizing 38 training views from GigaHands, we are able to faithfully synthesize the test views, demonstrating the effectiveness of our method in capturing complex hand motions and interactions.

PSNR	SSIM	LPIPS
29.50	0.963	0.063

Table 7. Quantitative Evaluation of Image Quality on Synthesized Views.

**Quantitative Evaluations.** Table 7 provides quantitative evaluations of the synthesized test views. We measure the rendering quality for 15 randomly selected motion sequences from GigaHands using Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM)[33], and Learned Perceptual Image Patch Similarity (LPIPS)[41]. These metrics confirm the high fidelity and perceptual quality of our synthesized views, highlighting the effectiveness of our dataset in novel view synthesis.



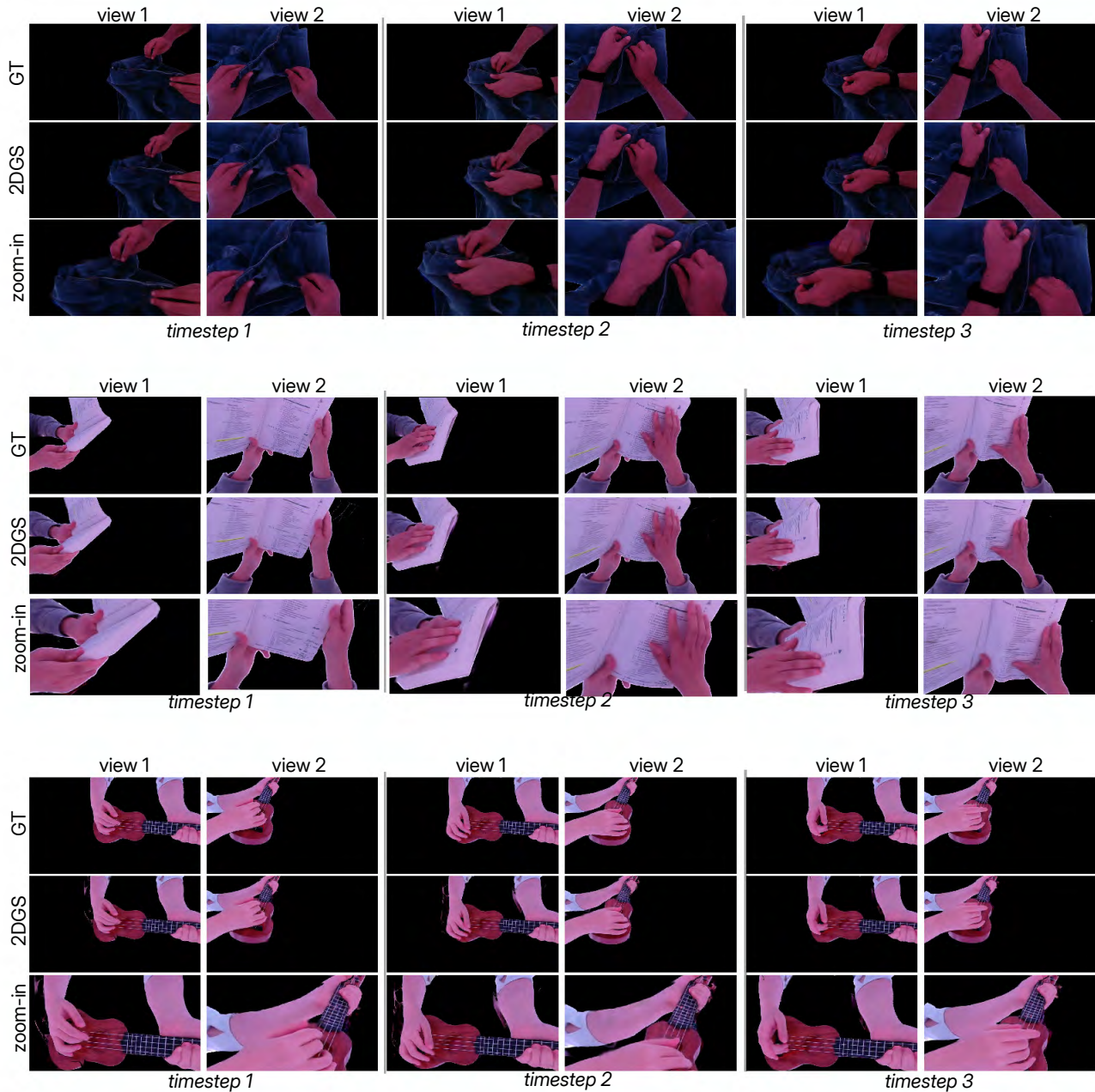


Figure 4. **Qualitative Results on Novel View Synthesis.**

## 6. Experiments on Hand-Object Interaction

To assess the effectiveness of our dataset in modeling hand-object interactions, we evaluate its applicability across two object-related tasks inspired by prior benchmarks in Oakink and TACO. GigaHands provides rich and diverse annotations of objects and their interactions with human hands, making it a valuable resource for learning and predicting hand-object dynamics. The selected tasks aim to demonstrate the advantages of GigaHands in capturing fine-grained motion patterns and enhancing downstream applications.

**Task-aware Motion Fulfillment.** The Task-aware Motion Fulfillment task aims to generate realistic hand motion sequences that align with predefined object trajectories while adhering to a given textual task description. To achieve this, we adapt the

Table 8. Quantitative results for text-driven motion synthesis with models trained on different datasets. *upper bound* indicates performance calculated with the ground truth. We report the mean of 20 evaluations, and  $\rightarrow$  means the closer to the upper bound the better. The model trained on GigaHands performs best on most metrics.

Dataset	R Precision(%) $\uparrow$			MM Dist. $\downarrow$	FID $\downarrow$	Div. $\rightarrow$	MM. $\uparrow$
	@1	@2	@3				
upper bound	50.4	71.2	81.1	3.67	0.022	9.30	-
OakInk2 [39]	23.4	35.7	49.8	7.41	13.1	6.24	3.71
upper bound	77.4	88.8	91.3	2.96	0.002	11.9	-
GigaHands	<b>27.2</b>	<b>46.2</b>	<b>54.6</b>	<b>6.12</b>	<b>5.91</b>	<b>10.2</b>	<b>9.73</b>

MDM baseline model [35] by incorporating dual-conditioning on both textual instructions and object trajectories. Given a task description and a sequence of object movements, the model predicts a corresponding sequence of hand motions that naturally interact with the object. For evaluation, we employ feature extractors and metrics used in text-to-motion tasks above. Table 8 provides a quantitative comparison between our approach and Oakink2. Our method outperforms Oakink2 across all evaluated metrics, highlighting its effectiveness in generating task-aware hand-object interactions.

Method	$J_e(mm, \downarrow)$	$T_e(mm, \downarrow)$	$R_e(^{\circ}, \downarrow)$
TACO	71.7 / 58.4	52.8	73.2
GigaHands	69.3 / 62.3	47.6	67.5

Table 9. Quantitative comparison of hand-object motion forecasting across different datasets. The evaluation metrics include Mean Per Joint Position Error ( $J_e$ ) for right / left hand pose prediction, translation error ( $T_e$ ) and rotation error ( $R_e$ ) for object motion. Lower values indicate better performance.

**Generalizable Hand-object Motion Forecasting.** Hand-object motion forecasting aims to predict future hand and object motions based on a short observed sequence. Given the poses of both hands and objects over  $N$  consecutive frames, the objective is to forecast their poses over the subsequent  $M$  frames. In our experiments, we set  $N = 10$  and  $M = 10$ . We adapt the MDM model by conditioning it on the past  $N$  frames of hand and object poses to predict their future states over the next  $M$  frames. Following human-object forecasting evaluations [18, 38], we assess hand pose estimation using Mean Per Joint Position Error  $J_e$ , while object motion is evaluated with translation error  $T_e$  and rotation error  $R_e$ . For TACO’s triplet representation, we condition only on tool poses and compute the corresponding metrics. Table 9 presents a comparative analysis of models trained on different datasets. While both models achieve comparable performance in hand pose prediction, our approach outperforms TACO in object motion forecasting. However, hand-object motion forecasting still remains a challenging task. The inherent complexity and rapid variations in generative motion patterns make modeling motion distributions difficult. Future work could explore more effective strategies to address this challenge.

## 7. Hand Motion Tracking

box detection	2D keypoints detection	time(150frames)	valid rate
detectron2	VITPose	6.5min	91.7%
yolov9c/track	VITPose	3.3min	89.6%
yolov9c/detect batch	VITPose	1.1min	85.4%
yolov9c/detect batch	HaMeR	3.0min	97.9%

Table 10. Comparison of different bounding box detection and 2D keypoint estimation methods.

Figure 5 illustrates the complete pipeline for hand motion tracking. The process begins with view-wise hand detection, tracking, handedness classification, and 2D keypoint extraction. Using multi-view 2D keypoints for both hands, we then triangulate each hand separately to obtain 3D hand keypoints. Finally, with the 2D keypoints and triangulated 3D keypoints,

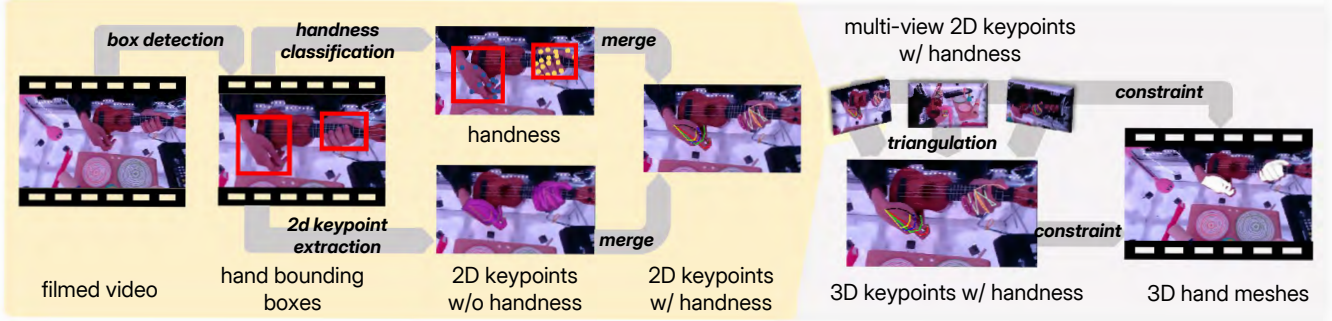


Figure 5. **Hand Motion Tracking Pipeline.**

we fit the MANO [32] parameters under these constraints. To validate our choices for hand detection, handedness classification, and 2D keypoint estimation, we calculate the number of valid 3D triangulated frames. We consider a 3D frame valid if, for both hands: (1) there are no missing keypoints; (2) the hand kinematics are normal, with bone length variations across frames within a certain threshold; and (3) the root of the hand moves temporally consistently. Frames that do not meet these criteria are considered invalid 3D triangulated keypoints. Table 10 compares the valid frame rate among 60 motion clips and the inference time for processing 150 frames (5 seconds).

**Hand Detection and Tracking.** For hand detection, we choose YOLO-v9 [30] as the backbone because it provides reliable detection results, runs efficiently compared to other backbones such as Detectron2[37], and yields consistent temporal results. Instead of using box tracking, we batch frames across multiple time steps (up to 256 images) and apply the detection function simultaneously, which accelerates inference time without significantly reducing the valid rate. During the filming process, we instructed subjects to keep both of their hands within the scene. Therefore, by default, we extract the two most confident bounding boxes labeled as 'hand' from the detection results.

**Handness (left or right).** To determine the handedness (left or right) of each detected hand, we follow the method adopted in HaMeR [27]. We use a side-aware checkpoint of ViTPose to detect keypoints within each bounding box. We classify a hand as 'right' if more than 60% of the detected keypoints correspond to the right hand, and 'left' if they correspond to the left hand.

**2D Keypoint Extraction.** We use HaMeR[27] for 2D keypoint extraction for two main reasons. First, it estimates a parametric representation of the hand from a cropped image, ensuring that all keypoints can be extracted from the output. Second, it provides more reliable results that ensure the hand kinematics are reasonable. We believe this is the primary reason that keypoints extracted from HaMeR lead to the most valid triangulated 3D keypoints. Although using HaMeR decreases inference speed, it significantly improves the valid rate.

**3D Keypoint Triangulation.** With the camera parameters and 2D keypoints extracted from multiple views, we triangulate the 3D keypoints for each hand. We remove outliers using RANSAC to improve accuracy.

**Parameter Fitting.** For MANO parameter fitting, we follow the EasyMoCap [1] pipeline, using both 2D and 3D keypoints as supervision. To capture fine-grained finger motions, we disable the PCA components and use a flat mean shape during the fitting process. This approach allows for more detailed and accurate hand mesh reconstruction.

## 8. Object Motion Tracking

The tracking process aims to estimate the 6DoF (six degrees of freedom) pose of the pre-scanned or generated mesh over time using multi-view RGB videos as constraints. Since our capturing system only contains RGB information without depth, pose estimation poses significant challenges. Figure 6 illustrates the complete object tracking pipeline. However, this can be mitigated by utilizing 51 camera views that provide 360-degree coverage of the scene. The tracking process generally consists of three stages: (1) extracting object masks across views as constraints; (2) providing a coarse pose estimation for

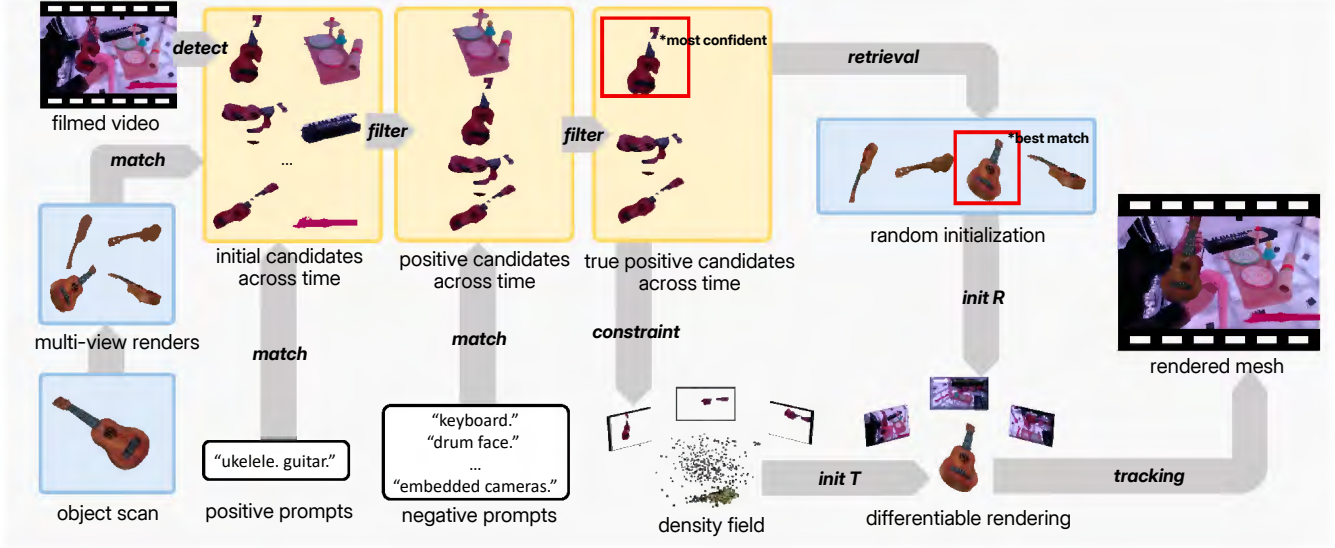


Figure 6. **Hand Motion Tracking Pipeline.**

stage	mask coverage
coarse estimation	45.2%
pose refinement (first frame)	91.1%
pose refinement (sequence)	78.5%

Table 11. Evaluation of Mask Coverage Rates Across Different Stages of Object Motion Tracking.

the mesh as an initialization for the following refinement process; (3) refining the object pose using multi-view pixel-level constraints to achieve accurate pose estimation. We use the mask coverage rate as an evaluation metric for tracking quality. The mask coverage rate is defined as the percentage of true positive segmented masks across frames that overlap with the rendered object’s silhouette during differentiable rendering. Table 11 provides this evaluation across different stages.

**Object Segmentation.** To use the multi-view RGB videos as constraints, accurate object masks are necessary. During the mask extraction process, we aim for three objectives: (1) obtaining masks from as many views as possible to guarantee more accurate results; (2) avoiding false positives in the segmentation masks, as they can be very distracting; (3) ensuring the masks are consistent across views to provide stable supervision. To achieve the first goal, for each view, we use DINOv2 [24] to detect as many object candidates as possible across time steps. We detect objects across time steps because an object might only appear during certain time period throughout the video. With multiple candidates that cover all objects across time steps, we address the second goal by using both image and text prompts to identify true positive candidates through feature matching methods using CLIP [23] and Grounding DINO [16]. Image prompts are generated by rendering object scans from random views on a sphere, while text positive prompts are manually designed. To remove false positives, we design negative prompts and compare them with the positive candidates. If an object candidate is more similar to the negative prompt than to the positive prompt, we regard it as a false positive. From all the true positive candidates, we select the most confident one using text-image similarity evaluated by CLIP [23]. With the most confident candidate, we can track the object across the video using SAM2 [29].

**Coarse Pose Estimation** Given the segmented object masks across views, we can initialize the coarse pose of the object. We first find the time step with the largest summed mask area of views. We use the multi-view masks at this time step to reconstruct the density field using Instant-NGP [21]. We threshold the reconstructed density field and find the largest cluster inside the field, shown as the yellow points in Figure 6. We then use the center of this density field as the coarse translation of the object. If the density field reconstruction fails, we use the center of the capture system with five random offsets as the coarse translations. Since the reconstructed density field is noisy, we cannot reliably use ICP (Iterative Closest Point) to



obtain the coarse rotation. Additionally, the geometric symmetry of the meshes requires us to use appearance information to initialize the rotation. We follow FoundPose [25] by randomly rotating the template mesh and retrieving the top five matched rotations using DINOv2 features. We use these as the initialization for rotation. Table 11 row 1 reports the mask coverage of the coarse estimation at this stage.

**Pose Refinement.** Given the coarse estimation of translation and rotation, we refine the pose estimation using differentiable rendering supervised with multi-view silhouette loss. The refinement consists of three stages. In the first stage, we select the best coarse initialization. In the previous step, we might have multiple translation and rotation candidates due to density field reconstruction failure and ambiguity in object symmetry. Therefore, we first optimize the pose starting from multiple coarse pose candidates for 200 steps. In the second stage, we select the coarse pose candidate with the minimal silhouette loss and further optimize the pose for 500 steps. Table 11 row 2 reports the mask coverage of the pose refinement for this frame after these two stages. In the third stage, we optimize the whole sequence, using the frame from the previous stage as initialization. Note that this time step is selected based on the maximal mask area, so it could be in the middle of a sequence. If this is the case, we need to optimize the loss temporally in both directions. For each next time step to optimize, we use its nearest optimized time step as initialization. We optimize 500 steps for each frame. Table 11 row 3 reports the final mask coverage across the sequence. Note that the mask coverage drops compared to the first frame; this is because for sequences with fast motion or where the object is severely occluded due to manipulation, the pose might lose track and accumulate errors.

## 9. Text Instructions and Annotations

In this section, we provide the prompts and examples used to create the instruction scripts and for annotation augmentation, along with details about the annotation interface.

### 9.1. Prompts and Examples for scenario grouping

In the scenario grouping phase, we start with verb pools extracted from multiple hand datasets. Our goal is to find objects corresponding to these verbs within various scenarios, enabling us to act out the associated actions. We prompt the LLM with specific instructions to generate verb-object pairs. From the LLM’s outputs, we select reasonable verb-object pairs that can be filmed on a tabletop setting. Here is an example to find objects associate with the verb ‘beat’.

Task: Using the verb "beat," generate a list of objects that can be acted upon with this verb in various scenarios. Ensure the objects are relevant and specific to the given context.

Scenarios

1. Cooking: Identify objects or ingredients commonly associated with "beat" in culinary activities.
2. Entertainment: List objects or tools that can be "beat" in entertainment or recreational contexts.
3. Housework: Suggest items or surfaces that are "beat" during cleaning or household chores.
4. Crafting: Find materials or tools that involve "beat" as part of a creative or crafting process.
5. Office Work: Consider any metaphorical or literal uses of "beat" with objects in an office or work environment.

Requirements:

1. Ensure that each object aligns with the context of the scenario.
2. Provide a diverse and creative range of examples for each scenario.
3. Be specific about the relationship between "beat" and the object.

### 9.2. Prompts and Examples for Scene structuring

In the scene structuring phase, we have already associated each scene with verbs and objects, organizing them through activities. However, in the manually designed raw activity scripts, some verbs related to the activities might be missing. To address this, we provide the LLM with a prompt to add the missing verbs to the scenes. For example below, we augmented

the "playing cards" part within the "Playing Monopoly, Cards, Coins, and Knucklebones" scene. After obtaining the output from the LLM, we perform a sanity check to ensure accuracy and coherence.

**\*\*Refined Prompt for Playing Cards Script\*\***

You are tasked with refining a script for playing cards. The script must follow the format: **\*\*[verb-ing]: detailed description\*\***. Several actions are missing, and you need to incorporate them into the sequence of actions in the correct position while ensuring the script is clear, organized, and detailed enough for a hand actor to act out.

**Instructions:**

1. Follow the format: Each action is described with a verb ending in "-ing" and its detailed description.
2. Include the following missing actions: scatter, slide, split, swap, wave.
3. Ensure the scenario is consecutive: The sequence of actions should flow logically without gaps.
4. Provide detailed hand action descriptions: Describe how the hands move, grip, and interact with the cards for clarity and precision.
5. This script is for a hand actor to act out. The quality of your addition and organization will determine the final output.
6. Ensure the sequence of actions forms a coherent and consecutive scenario.

If you revise the script well, I will reward you \$20.

---

**Original Script:**

**Play Cards**

- [Bridge Shuffling]: Hold the deck with your right hand. Use your thumb and middle finger to grip the deck on the short sides, with your index finger resting along the long edge of the deck. Do the bridge shuffle.
- [Regular Shuffling]: Shuffle the deck of cards by interweaving them with your hands, mixing them thoroughly.
- [Cutting + Dealing]: Cut the deck of cards in half using a quick motion, separating it into two smaller decks. Deal the cards to the players one by one, distributing them evenly.
- [Flipping]: Flip the top card of the deck face-up, revealing its value or suit.
- [Fanning + Checking + Sorting]: Fan out the cards in your hand, creating a spread of cards that can be easily viewed. Check the value of your cards by looking at them without revealing them to others. Sort the cards in your hand according to their suits or numerical order.
- [Drawing + Discarding]: Draw a card from the deck and add it to your hand, increasing the number of cards you hold. Discard a card from your hand by placing it face-down on a designated discard pile.
- [Collecting + Stacking]: Collect the cards from all players after a round of the game has ended. Stack the cards on top of each other, forming a neat pile.

---

### 9.3. Prompts and Examples for instruction scripting

In the instruction scripting phase, we already have an activity script for each scene, where each activity is associated with a list of verbs that occur sequentially. Our goal here is to expand each verb into a complete instruction that helps fulfill the activity. We provide the LLM with a prompt to achieve this expansion. For instance below, we applied this process to an activity in the "Making and Drinking Tea" scene. After receiving the output from the LLM, we conducted three rounds of sanity checks to ensure the instructions were accurate and coherent.

You are tasked with refining and structurizing the given hand action script. The final script must follow these guidelines:

1. Format: Each action must follow the format `[verb-ing]: detailed description of the action.`
2. Retain All Verbs: Do not delete or remove any verbs provided in the brackets.
3. Separate Multiple Verbs: If a bracket contains multiple verbs, split them into individual actions, each with its own description.
4. Expand Missing Verbs: If a verb is implied but not explicitly described in the action, add it with an accurate and detailed description.
5. Verb Format: Change all verbs into the present participle format (`-ing` form).
6. Clarity and Detail: Ensure the descriptions are clear, precise, and detailed enough for a hand actor to perform the actions.

---

Example:

Original Action:

`[grip, set, lift]: Grip the teapot lid with the right hand, lift it, and set it aside.`

Refined Actions:

- [Gripping]: Grip the teapot lid firmly with the right hand.
- [Lifting]: Lift the teapot lid straight up, keeping it steady.
- [Setting]: Set the teapot lid down gently on a flat surface.

### 9.4. Annotation Interface

For the annotation phase, we already have the instruction scripts and the motion sequences filmed accordingly. Our task is to annotate any actions that were not mentioned in the original instruction scripts. Figure 7 illustrates the annotation interface we used. This interface loads the filmed multi-view videos with the hand mesh rendered on top for visual clarity. The original instruction is displayed below the video. Features such as a progress bar and controls like "Play," "+1 Frame," and "-1 Frame" buttons are provided to accurately segment the motion sequence. Additionally, buttons like "Match Annotation," "Add Annotation," "Remove Annotation," "Split Video," and "Remove Split" are available to correct or refine the original instructions.

### 9.5. Prompts and Examples for text augmentation

In the text augmentation phase, we have the motion clips along with their text annotations. To enhance the diversity of descriptions, we aim to augment these annotations since one action can be described in multiple ways. We feed the LLM with a specific prompt for text augmentation. Outputs such as system errors or phrases like "I'm sorry" are removed to maintain data quality and consistency.

You are a sentence rewriter. Your task is to rewrite the provided input sentence into five different variations. You can vary the verbs and descriptions but **\*\*do not add any new actions or change the original meaning\*\***.

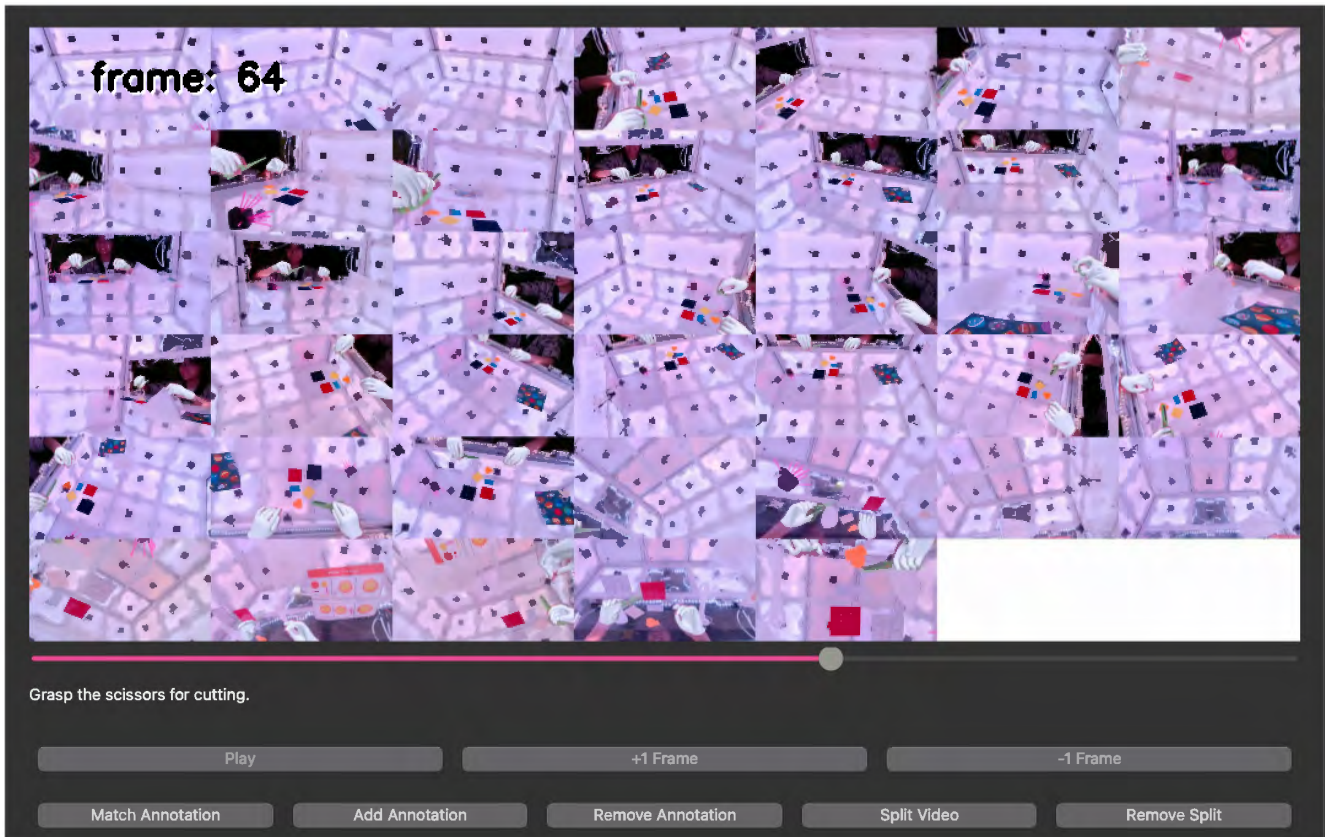


Figure 7. **Annotation Interface.**

Instructions:

- Begin the output with: "Rewritten Sentences:"
- Each sentence should be separated by the symbol "\$"
- Use different verbs or phrasing to achieve natural, varied expressions without changing the action or intent of the original sentence.

Example:

Input: "Take out an egg from the carton.\n"

Output: Rewritten Sentences: Remove an egg from the carton. \$ Pick out an egg from the carton. \$ Pull an egg from the carton. \$ Grab an egg from the carton. \$ Lift an egg from the carton.

## 10. Dataset Inspection

### 10.1. Object Visualization.

Figure 8 shows 96 of the objects in GigaHands by randomly select a few objects each scene. The objects spans diverse scenarios and functionalities.





Figure 8. Randomly Sampled Objects.

## 10.2. Verb Pool.

GigaHands contains a total of 1467 verbs. Table 12 and Tab. 13 shows the most frequent and least frequent verbs in the text annotation.

place	take	use	put	press	set	open	remove
position	move	hold	pull	seize	apply	pick	lift
secure	slide	grab	lay	turn	grasp	tap	push
extract	clutch	insert	shift	detach	fasten	adjust	grip
rest	twist	rotate	arrange	snatch	hit	spread	return
close	retrieve	employ	release	loosen	squeeze	drop	flip
glide	rub	make	click	fold	attach	touch	clean
cut	get	leave	raise	sweep	wipe	switch	collect
shut	seal	transfer	cover	snap	draw	withdraw	keep
spin	swipe	select	separate	strike	perform	shake	toss
bring	utilize	pour	extend	activate	compress	stretch	smooth
pinch	ease	execute	unseal	pluck	tighten	lock	dab
gather	uncap	throw	scoop	roll	break	create	hoist
give	brush	flick	change	reduce	wrap	elevate	fill
trim	clasp	work	engage	distribute	swivel	slice	increase
drag	deposit	slip	fetch	unfold	split	let	flatten
pat	pass	fit	add	divide	bend	peel	dispense
decrease	mix	deliver	connect	agitate	trace	organize	choose
undo	stick	form	guide	gesture	clip	continue	swing
play	blend	disconnect	tear	stir	wiggle	carry	tilt
alter	direct	affix	coat	run	modify	unzip	uncover
clench	dry	shape	present	mark	maintain	depress	sketch
slow	massage	speed	lessen	replace	rip	wind	join
reach	submerge	unplug	wave	capture	immerse	bind	reveal
point	crush	accelerate	diminish	fix	swirl	straighten	navigate
snip	carve	ensure	restore	free	strip	combine	unroll
obtain	handle	store	tie	wash	sway	repeat	strum
intensify	lower	disengage	aim	plug	chop	quicken	unwind
catch	jiggle	reverse	go	thread	revolve	stroke	identify
display	pound	jostle	render	highlight	coil	knock	pretend
stack	relocate	scroll	untangle	rinse	show	amplify	clap
beat	start	enhance	wrench	spoon	tidy	align	pop
zip	acquire	unravel	shave	punch	widen	indicate	enter
write	dump	assemble	remover	launch	loop	knot	crack
curl	encircle	weave	plunge	do	opt	soak	suspend
enclose	lengthen	examine	drizzle	dispose	fle	sprinkle	pump
expand	unhook	grind	send	build	clamp	nudge	smack
relax	operate	stow	pack	hasten	tickle	shear	buckle
interlock	whisk	act	dunk	lean	stop	hurl	shuffle
hand	caress	flex	envelop	slot	drive	shoot	introduce

Table 12. Most Frequent 320 Verbs.

### 10.3. List of Scenarios and Scenes.

Table 14 presents all of the scenarios and scenes in GigaHands.

## References

- [1] Easymocap - make human motion capture easier. Github, 2021. [11](#)
- [2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Al-tenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. [6](#)
- [3] Prithviraj Banerjee, Sindi Shkodrani, Pierre Moulon, Shreyas Hampali, Fan Zhang, Jade Fountain, Edward Miller, Selen Basol,

hydrate	nourish	gain	closed	travel	document	interchange	submit
disorganize	investigate	convert	decorate	opener	imperfect	realign	be
disarrange	toggle	tab	purify	enfold	seek	tension	container
crisscross	come	systematize	chuck	tangle	slurp	bunch	relay
stem	filter	wield	impact	protect	achieve	slam	narrow
remain	dish	null	trickle	segregate	distinguish	rely	assist
disjoin	fuse	fish	disintegrate	heave	patch	file	waver
buff	stab	excise	shoelace	swell	practice	uncrease	notch
avoid	estimate	confirm	probe	unlink	disassemble	underline	unbend
orient	crash	haul	redo	enact	tense	discharge	disburse
amass	annotate	state	incorporate	fire	rid	reproduce	modulate
diffuse	collide	assort	invite	bash	beetle	blower	wound
lash	mount	swish	actuate	doodle	help	suck	absorb
center	revert	need	ascertain	demonstrate	lodge	deflate	shove
delve	repack	stuff	wriggle	cascade	stri	imbibe	encapsulate
freshen	back	scrap	sieve	screen	ignite	ajar	cool
swinge	defend	exhale	trill	uptick	augment	cuddle	inspire
exame	belt	srew	sit	quench	nil	isolate	copy
overcross	authenticate	reaccess	register	poke	flood	calm	remit
burrow	bottle	engrave	glaze	speckle	jitter	strain	stash
encompass	commence	erase	structure	delineate	envelope	pamper	neaten
mention	designate	jump	muster	signify	aid	distill	swarm
overflow	infest	glitch	unblock	rise	weigh	hover	amp
categorize	establish	hinge	emphasize	specify	assert	rebuild	seem
fragment	saw	retie	conclude	finalize	scribe	acknowledge	portray
tumble	perfect	reinforce	temper	define	flaunt	amalgamate	tend
recompose	customize	water	irrigate	coax	uplift	magnify	grap
sud	prompt	jingle	aside	leftover	peal	infuse	nest
bury	quarter	mop	cap	heft	nearer	expel	dress
interleave	cooperate	crawl	intermingle	rummage	briskly	hurry	float
reform	nab	bake	amend	shade	unsecure	harvest	discard
sample	recite	advise	grace	drum	repress	reapply	rectify
hop	leverage	ring	substitute	issue	believe	channel	object
tease	crease	snuff	draft	survey	bounce	test	hack
pipe	refit	prefer	style	duplicate	echo	shorn	fluctuate
twitch	pressurize	toast	melt	latticework	twiddle	impress	disturb
tremble	recreate	snug	envision	think	interweave	redirect	paste
scald	permit	tip	interconnect	improve	experience	refill	stock
resort	impart	involve	claw	pry	drap	chip	hone
allot	deck	trap	juggle	clarify	retreat	unfurl	disband

Table 13. Least Frequent 320 Verbs.

Richard Newcombe, Robert Wang, et al. Introducing hot3d: An egocentric dataset for 3d hand and object tracking. *arXiv preprint arXiv:2406.09598*, 2024. 3

- [4] Junuk Cha, Jihyeon Kim, Jae Shin Yoon, and Seungryul Baek. Text2hoi: Text-guided 3d motion generation for hand-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1577–1585, 2024. 8
- [5] Zicong Fan, Omid Taheri, Dimitrios Tzionas, Muhammed Kocabas, Manuel Kaufmann, Michael J Black, and Otmar Hilliges. Arctic: A dataset for dexterous bimanual hand-object manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12943–12954, 2023. 3, 8
- [6] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF*

scenarios	scenes
Cooking, Cleaning, Eating.	Making and Drinking Tea Making, Eating and Cleaning Instant Noodle Eating and Packing Delivered Food Making an open egg and spam sandwich Making a Bowl of Salad Baking and Frosting a Bread
Office Working	Drawing Mind Map Receiving and Sending a File Testing the laptop Testing Tablet DigitalPen Phone SmartWatch on Charging Station Filming gestures with GoPro Unwrapping and Wrapping a present
Crafting	Seal Carving and Smoothing Assemble and Disassemble Kids Tool Bench Cultivating and uproot a Plant Sewing kit, Crocheting, Knitting and Finger Knitting
Entertaining	Playing monopoly, cards, coins and knucklebones Playing mini instruments Desktop Boxing Paying with a Puppy Dog
Houseworks	Applying Makeups Packing a gym bag Massaging Your Own Hands First Aid Cleaning Glass Tabletop

Table 14. All 5 Scenarios and 25 Scenes.

*Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022. 3

- [7] Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. Ego-exo4d: Understanding skilled human activity from first-and third-person perspectives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19383–19400, 2024. 3
- [8] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5152–5161, 2022. 3, 4
- [9] Chuan Guo, Xinxin Zuo, Sen Wang, and Li Cheng. Tm2t: Stochastic and tokenized modeling for the reciprocal generation of 3d human motions and texts. In *ECCV*, 2022. 4, 5, 6
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3
- [11] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017. 3
- [12] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 8
- [13] Juntao Jian, Xiuping Liu, Manyi Li, Ruizhen Hu, and Jian Liu. Affordpose: A large-scale dataset of hand-object interactions with affordance-driven hand pose. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14713–14724, 2023. 3
- [14] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*, 2015. 7
- [15] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004. 7
- [16] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 12



- [17] Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21013–21022, 2022. [3](#)
- [18] Yun Liu, Haolin Yang, Xu Si, Ling Liu, Zipeng Li, Yuxiang Zhang, Yebin Liu, and Li Yi. Taco: Benchmarking generalizable bimanual tool-action-object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21740–21751, 2024. [3](#), [6](#), [10](#)
- [19] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. [3](#)
- [20] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. [3](#)
- [21] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. [12](#)
- [22] Takehiko Ohkawa, Kun He, Fadime Sener, Tomas Hodan, Luan Tran, and Cem Keskin. Assemblyhands: Towards egocentric activity understanding via 3d hand pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12999–13008, 2023. [3](#)
- [23] Abby O’Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023. [12](#)
- [24] 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. [12](#)
- [25] Evin Pinar Örnek, Yann Labbé, Bugra Tekin, Lingni Ma, Cem Keskin, Christian Forster, and Tomas Hodan. Foundpose: Unseen object pose estimation with foundation features. *arXiv preprint arXiv:2311.18809*, 2023. [13](#)
- [26] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002. [7](#)
- [27] Georgios Pavlakos, Dandan Shan, Ilija Radosavovic, Angjoo Kanazawa, David Fouhey, and Jitendra Malik. Reconstructing hands in 3d with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9826–9836, 2024. [11](#)
- [28] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. [3](#)
- [29] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. [12](#)
- [30] Dillon Reis, Jordan Kupec, Jacqueline Hong, and Ahmad Daoudi. Real-time flying object detection with yolov8. *arXiv preprint arXiv:2305.09972*, 2023. [11](#)
- [31] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 2017. [3](#)
- [32] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *arXiv preprint arXiv:2201.02610*, 2022. [11](#)
- [33] Umme Sara, Morium Akter, and Mohammad Shorif Uddin. Image quality assessment through fsim, ssim, mse and psnr—a comparative study. *Journal of Computer and Communications*, 7(3):8–18, 2019. [8](#)
- [34] Tianxiao Shen, Myle Ott, Michael Auli, and Marc’Aurelio Ranzato. Mixture models for diverse machine translation: Tricks of the trade. In *International conference on machine learning*, pages 5719–5728. PMLR, 2019. [7](#)
- [35] Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. Human motion diffusion model. In *The Eleventh International Conference on Learning Representations*, 2023. [4](#), [5](#), [10](#)
- [36] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. [3](#)
- [37] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019. [11](#)
- [38] Sirui Xu, Zhengyuan Li, Yu-Xiong Wang, and Liang-Yan Gui. InterDiff: Generating 3d human-object interactions with physics-informed diffusion. In *ICCV*, 2023. [10](#)
- [39] Xinyu Zhan, Lixin Yang, Yifei Zhao, Kangrui Mao, Hanlin Xu, Zenan Lin, Kailin Li, and Cewu Lu. Oakink2: A dataset of bimanual hands-object manipulation in complex task completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 445–456, 2024. [3](#), [6](#), [10](#)
- [40] Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Yong Zhang, Hongwei Zhao, Hongtao Lu, Xi Shen, and Ying Shan. Generating human motion from textual descriptions with discrete representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14730–14740, 2023. [3](#), [4](#), [5](#)
- [41] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. [8](#)
- [42] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5745–5753, 2019. [5](#)