

SPLite Hand: Sparsity-Aware Lightweight 3D Hand Pose Estimation

Yeh Keng Hao*

National Tsing Hua University
Taiwan
haoyeh@gapp.nthu.edu.tw

Hsu Tzu Wei†

National Tsing Hua University
Taiwan
welly916@gmail.com

Sun Min‡

National Tsing Hua University
Taiwan
sunmin@ee.nthu.edu.tw

Abstract

With the increasing ubiquity of AR/VR devices, the deployment of deep learning models on edge devices has become a critical challenge. These devices require real-time inference, low power consumption, and minimal latency. Many framework designers face the conundrum of balancing efficiency and performance. We design a light framework that adopts an encoder-decoder architecture and introduces several key contributions aimed at improving both efficiency and accuracy. We apply sparse convolution on a ResNet-18 backbone to exploit the inherent sparsity in hand pose images, achieving a 42% end-to-end efficiency improvement. Moreover, we propose our SPLite decoder. This new architecture significantly boosts the decoding process's frame rate by 3.1× on the Raspberry Pi 5, while maintaining accuracy on par. To further optimize performance, we apply quantization-aware training, reducing memory usage while preserving accuracy (PA-MPJPE increases only marginally from 9.0 mm to 9.1 mm on FreiHAND). Overall, our system achieves a 2.98× speed-up on a Raspberry Pi 5 CPU (BCM2712 quad-core Arm A76 processor). Our method is also evaluated on compound benchmark datasets, demonstrating comparable accuracy to state-of-the-art approaches while significantly enhancing computational efficiency.

CCS Concepts

- Computing methodologies → Convolutional neural networks; Computer vision; Machine learning.

Keywords

Hand-object interaction, Hand Pose estimation, Sparse Convolution, Machine Learning

ACM Reference Format:

Yeh Keng Hao, Hsu Tzu Wei, and Sun Min. 2025. SPLite Hand: Sparsity-Aware Lightweight 3D Hand Pose Estimation. In *Proceedings of The 2025 Annual International Conference on Cognitive Computing (AICCC) (AICCC)*. ACM, New York, NY, USA, 10 pages. <https://doi.org/XXXXXXX.XXXXXXX>

*First author and primary contributor.

†Second author.

‡Third author. Project advisor.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

AICCC, Tokyo, JP

© 2025 ACM.

ACM ISBN 979-8-4007-1889-2

<https://doi.org/XXXXXXX.XXXXXXX>

1 Introduction

3D hand pose estimation from a single 2D input is vital for applications in robotics, VR, and AR. The primary challenge is balancing high accuracy with computational efficiency, especially for devices with limited resources.

Rather than designing transformer-based architectures [1–3] to achieve higher accuracy, recent research has focused on building cost-effective, lightweight neural networks with reduced computational complexity so as to deploy on edge devices. Works such as MobileNet [4, 5], ShuffleNet [6], and GhostNet [7] introduced efficient operators such as depthwise and group convolution. However, these methods often suffer from increased memory access costs and suboptimal memory utility.

A common pipeline for single-view hand pose estimation typically consists of three stages: 2D encoding, 2D-to-3D feature lifting, and 3D decoding. Most of the computation occurs during the 2D encoding phase. Lin *et al.* [8] directly use MobileNet to achieve a higher frame rate [4, 5], which struggles with predicting occlusion and complex scenarios. It also can't apply custom operations like sparse convolution, while others [9, 10] introduce customized blocks to extract 2D hand keypoints $J_{2D} \in \mathbb{R}^{21 \times 2}$. Over time, several techniques have been proposed to make deep networks more compact, such as pruning [11, 12], low-bit quantization [13, 14], and knowledge distillation [15, 16]. However, the effectiveness of these methods is often upper-bounded by the quality of the pre-trained models used as baselines.

To reduce redundancy in feature maps and improve efficiency, Chen *et al.* [17] proposed a heuristic partial-channel design, FasterNet, which operates only on selected channels. Inspired by this, we adopt a 3-stage architecture—2D encoder, 2D-3D feature lifting, and 3D decoder, termed SPLite, to predict both 3D keypoints and mesh as illustrated in Figure 1.

In the 2D encoding stage, we use a proprietary algorithm to generate the edge image, as shown in Figure 3, and send it to the early-fusion block, which fuses two types of edge modalities. This avoids the need for modifying the backbone while naturally introducing sparsity. Sparse convolution, unlike dense convolution, operates only on non-zero elements, reducing both memory usage and computational load. Inspired by Zhang *et al.* [18], we leverage sparse convolution to focus on high-intensity edge regions, improving inference speed by 42% without sacrificing accuracy. This is particularly beneficial for deploying the model on edge devices with limited power consumption.

For the 2D-to-3D lifting stage, we adopt a simplified pose-to-vertex module based on Chen *et al.*'s lifting module [19], which uses a lifting matrix to project the 2D sequential feature maps into 3D representations.

In 3D decoders, *Graph Convolutional Networks* (GCNs) [20, 21] are a specialized type of neural network that can effectively analyze irregular data structure, such as meshes. GCNs are often used to handle these data types by learning meaningful representations for each node. GCNs achieve this by aggregating and combining features from a node’s neighbors, which allows them to capture both the node’s individual properties and its structural context within the graph.

Although many researchers have proposed improvements to this operation—for example, Gong *et al.* proposed SpiralNet++ [22] to enhance SpiralNet [23]—these new architectures often slow down the forward pass because the vertex sampling process is computationally expensive.

Large pre-trained networks are effective for vision tasks but face memory and computation limitations on embedded systems. While model quantization to low-bit precision can compress these networks, it often sacrifices accuracy. Wang *et al.* [13, 24] have shown that Quantization-aware training (QAT) can preserve accuracy. We first train a floating-point baseline, then apply QAT, and convert the model to ONNX for deployment. An adjustable learning rate compensates for reduced accuracy. On benchmark dataset[25], our quantized model reduces size by 75% (from 72MB to 18MB) with only a 0.1mm drop in PA-MPJPE (from 9.0mm to 9.1mm).

Finally, we introduce a new multimodal hand-object interaction dataset with dual-view perspectives. It includes RGB, grayscale, and edge modalities across 100 object categories and manipulation actions.

Our key contributions

- We introduce a sparse data transformation pipeline and apply sparse convolution to our encoder, achieving a 42% speed-up.
- We propose the **SPLite decoder**, a hardware-friendly graph-based decoder. It accelerates inference by up to 3.1× (vs. *MobRecon-tailored* [19]) and 65% (vs. *ResNet18* [26]) on Raspberry Pi 5.
- We apply quantization-aware training to compress our model from 72MB to 18MB ($\downarrow 75\%$), with only a 0.1mm accuracy drop in PA-MPJPE.
- We present a diverse and challenging hand-object interaction dataset, featuring multi-modal data across 100 object categories and manipulation actions for audience to evaluate their models.

2 Related Work

Lightweight Networks

Lightweight architectures have been extensively studied to reduce computational burden, particularly for deployment on resource-constrained platforms. MobileNet [4, 5] utilizes depthwise separable convolutions (depth-wise and pointwise convolutions) to reduce computation. ShuffleNet [6] introduces a channel shuffle mechanism to enhance information flow and reduce latency. GhostNet [7] further leverages depthwise and group convolutions to extract spatial features more efficiently. Chen *et al.* [19] proposed a lightweight model design as our baseline. After replacing the encoding and decoding modules of our baseline model with a customized version, we achieved a significant breakthrough across several performance

metrics. While these methods have demonstrated effectiveness in reducing computation, many of them incur increased memory access costs and inefficient memory utility—limitations that become critical on edge devices. Moreover, many “lightweight” models are only validated on high-end GPUs, rather than actual edge platforms. In contrast, our SPLite operator shows efficiency and memory improvements on real-world edge hardware such as the Raspberry Pi 5, and even outperforming baselines in several use cases, as shown in Figure 2.

Sparse Convolution

Convolutional neural networks (CNNs) have achieved significant success across various visual tasks, including 2D/3D hand pose estimation. However, the dense nature of traditional convolution operations leads to high computational overhead, especially for high-dimensional outputs like 3D meshes. To address this, libraries such as Minkowski Engine [27] and TorchSparse [28] have been developed to enable efficient sparse convolution for 3D point cloud processing. In 2D domains, Hsieh *et al.* [29] introduced sparse convolution for body pose estimation by leveraging motion vector cues. However, this approach is ineffective when the subject is still, as motion vectors provide no useful information. To overcome this limitation, we adopt a different fusion strategy and incorporate an accelerating module ,termed SPLite, within our decoding process. Inspired by these works, we propose to process input images into edge-enhanced sparse representations and then apply sparse convolution. Our experiments show that when the input sparsity reaches a mean of 86%, our sparse convolution design improves inference speed by 42%.

Hand-Object Interaction Datasets

Hand-object Interaction (HOI) datasets play a central role in training models for realistic manipulation tasks. Several well-known datasets have been introduced, such as HO-3D [30], HOI4D [31], and HOT3D [32], providing benchmarks for joint hand-object tracking and pose estimation. However, many of these datasets are limited in either object diversity, interaction complexity, or sensory modality.

Our dataset addresses these limitations by incorporating synchronized RGB, grayscale, and edge modalities, with over 100 object categories and manipulation types. It provides dual-view perspectives, enabling richer supervision and facilitating better generalization across interaction scenarios.

3 Our Method

We aim to accelerate our model without sacrificing accuracy. To achieve this, we design an efficient pipeline consisting of several stages: early fusion, 2D encoding, feature lifting, and 3D decoding. Figure 1 illustrates the holistic architecture. Our objective is to predict a 3D hand mesh from a monocular RGB image.

First, the input RGB image is converted to grayscale, and an edge map is produced using our proprietary edge-detection algorithm—methods such as the Canny[33] or Sobel detectors[34] yield comparably effective results. We employ a customized backbone, *Sparse ResNet-18*, which leverages the inherent sparsity of the input data by applying sparse convolutional blocks.

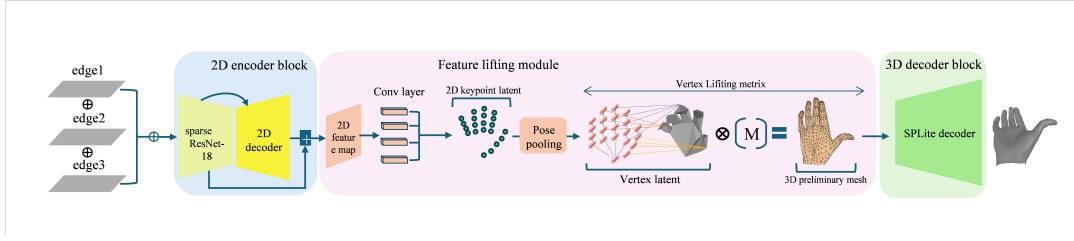


Figure 1: our efficient pipeline begins with an RGB image being converted to grayscale, from which a proprietary algorithm extracts an edge image. A customized backbone called Sparse ResNet-18 then uses sparse convolutional blocks to generate 2D feature maps and predicted depth values. To improve the robustness of the depth prediction, a second data stream is introduced during training to account for perspective ambiguity. Finally, a calibrated camera intrinsic transformation module converts the 2D feature maps from camera coordinates to real-world coordinates for the lightweight SPLite decoder, which ultimately predicts a 3D hand mesh.

In the 2D encoding stage, our network produces preliminary 2D feature maps in camera coordinates (u_i, v_i) , $i = 1 \dots N$, along with predicted depth values. Since monocular depth estimation is ill-posed due to perspective ambiguity, we enhance the robustness of depth prediction during training by introducing a second-view data stream.

The 2D feature maps are then transformed from camera coordinates into real-world coordinates using a calibrated camera intrinsic transformation module. This transformation ensures spatial consistency before feeding the features into our lightweight SPLite decoder.

3.1 Early Fusion

As shown in Figure 1, we first process the RGB input images into edge maps. The pixel values of these edge images range from 0 to 255, where higher values indicate stronger edge intensity. A fusion module then combines two types of edge representations into a sandwich-like structure, which is passed to the 2D encoding module.

3.2 2D Encoding

We adopt ResNet-18 as the backbone for our 2D encoder due to its lightweight architecture and widespread deployment on edge devices. The sparsity of our input data allows us to integrate Minkowski sparse convolutions [27], resulting in a 36%–42% speed-up on Intel® Core™ i9-9980XE CPU. Also, We leverage 7×7 kernel, a larger convolutional kernel size, to capture broader spatial hand pose context and enhance feature representation. On average, our dataset exhibits an input sparsity of 89.2%, enabling sparse convolution to deliver performance comparable to dense convolution while drastically reducing the inference time. This strategy has also proven successful in domains such as 3D point cloud processing and 2D character recognition.

3.3 2D-to-3D Feature Lifting

To lift 2D features into a 3D representation, we utilize a feature elevation component that connects the 2D and 3D realms via a sequence of optimized procedures. This component commences with heatmap-guided estimation to forecast 2D landmark locations

from the feature grid, proceeds to gather synchronized features. We associate it with a predicted depth value d , which allows us to compute the corresponding 3D point (x, y, z) in real-world coordinates. The process consists of the following steps:

- **Heatmap-regression:**

The 2D feature grid generated by the encoder is handled to estimate 2D landmark locations. This entails producing probability distributions over a mid-resolution feature grid

$$L_l = \{L_l^j\}_{j=1}^M,$$

where M is the keypoint count, 21 keypoints in a hand.

- **Pose Pooling:**

Leveraging the forecasted 2D landmark locations, we derive alignment-synchronized features from the encoded 2D feature grid F_g :

$$F_p = [F_g(L_l^j)]_{j=1}^M.$$

This produces a collection of M feature vectors $F_a \in \mathbb{R}^{M \times C_g}$, where C_g is the feature dimension, all spatially matched to the 2D landmarks.

- **Pose-to-Vertex Lifting:**

To convert the 2D-synchronized features into 3D mesh point features, we apply a trainable lifting matrix on a subsampled reference mesh. The mapping is executed as:

$$F_{\text{mesh}} = M \cdot F_p$$

where M denotes the trained Matrix that translates landmark features to subsampled mesh features, upholding conceptual and structural integrity without depending on overt spatial transformations.

This transformation embeds geometric consistency into the feature map, which is crucial for accurate 3D hand mesh reconstruction. The resulting 3D-aware features, now spatially aligned in real-world coordinates, are then passed to our SPLite decoder for further processing.

This transformation embeds geometric consistency into the feature map, which is crucial for accurate 3D hand mesh reconstruction. The resulting 3D-aware features, now spatially aligned in real-world coordinates, are then passed to our SPLite decoder for further processing.

3.4 3D Decoding with SPLite Decoder

We propose a novel SPLite graph operator inspired by Spiral Convolution [22], tailored for efficient hardware deployment. Traditional spiral-based graph convolutions suffer from high latency due to sequential vertex traversal. To overcome this, we introduce a parallelized sampling strategy that restructures the vertex indexing process to support SIMD (Single Instruction, Multiple Data) operations at the compiler level. This enables simultaneous processing of multiple vertex indices, accelerating vertex sampling process and improving CPU utilization.

Specifically, the SPLite module parallelly indexes the vertex latent, using only a quarter of the channels for the decoding convolution. This technique efficiently reduces redundant computation and memory access time while preserving model accuracy. These two features—parallel indexing and partial channel convolution are the foundation of our SPLite decoder.

Integrated together, these optimizations yield significant performance improvements. Specifically, our SPLite decoder achieves a 28% speed-up over the MobRecon-tailored [19] baseline, and a 65% speed-up compared to MobRecon [19] with ResNet-18 [26] on the Raspberry Pi 5 CPU. Ablation studies indicate that the partial channel decoding alone contributes up to a 2.84× speed increase over full-channel decoding.

4 Loss Functions

To train the model and ensure accurate 3D reconstruction, we employ various loss functions that optimize different aspects of the 3D pose and shape estimation process. These loss functions typically focus on the following objectives:

4.1 Reprojection Loss

We adopt reprojection loss to minimize the difference between the 2D projections of the estimated 3D keypoints and the actual 2D keypoints observed in the image. It is formulated as:

$$L_{\text{reproj}} = \sum_i \|\mathbf{P}_i^{\text{2D}} - \hat{\mathbf{P}}_i^{\text{2D}}\|_2^2$$

Where:

- \mathbf{P}_i^{2D} is the ground truth 2D position of the i -th keypoint.
- $\hat{\mathbf{P}}_i^{\text{2D}}$ is the predicted 2D position of the i -th keypoint, which is obtained by projecting the predicted 3D points into the 2D image space using the intrinsic matrix \mathbf{K} .

The goal of this loss is to reduce the Euclidean distance between the predicted and ground truth 2D points, ensuring that the projected 3D points match the 2D observations as closely as possible.

4.2 3D Pose Loss

After our model predict 3D keypoint position. 3D pose losses focus on the difference between the predicted 3D keypoints and the ground truth 3D keypoints. It is computed with the L2 norm:

$$L_{\text{pose}} = \sum_i \|\hat{\mathbf{P}}_i^{\text{3D}} - \mathbf{P}_i^{\text{3D}}\|_2^2$$

Where:

- \mathbf{P}_i^{3D} is the ground truth 3D position of the i -th keypoint.

- $\hat{\mathbf{P}}_i^{\text{3D}}$ is the predicted 3D position of the i -th keypoint.

The goal of 3D keypoint loss function is to minimize the error between the predicted 3D points and their ground truth counterparts.

4.3 Depth Loss

Estimating depth from a single RGB image is inherently error-prone. We obtain ground-truth depth data from the Intel® RealSense™ D435 camera. The depth loss is defined as:

$$L_{\text{depth}} = \sum_i \|d_i - \hat{d}_i\|_2^2$$

Where:

- d_i is the ground truth depth of the i -th keypoint.
- \hat{d}_i is the predicted depth of the i -th keypoint.

This loss directly affects the reconstruction of the 3D structure.

4.4 Smoothness Loss

To improve the smoothness of the reconstructed 3D mesh, a smoothness loss is applied to the 3D vertices, ensuring that neighboring vertices in the mesh are not overly distorted. It is formulated as:

$$L_{\text{smooth}} = \sum_i \|\hat{\mathbf{V}}_i - \hat{\mathbf{V}}_j\|_2^2$$

Where:

- $\hat{\mathbf{V}}_i$ and $\hat{\mathbf{V}}_j$ are neighboring vertices in the 3D mesh.

Smooth loss encourages spatial consistency between neighboring points, which helps in generating more realistic and smooth 3D meshes.

4.5 Aggregation Loss

The aggregation loss used for training is a weighted sum of the individual losses:

$$L_{\text{aggregation}} = \lambda_1 L_{\text{reproj}} + \lambda_2 L_{\text{pose}} + \lambda_3 L_{\text{depth}} + \lambda_4 L_{\text{smooth}}$$

Where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are hyperparameters that control the relative importance of each loss term.

5 Experiments

5.1 Experiment details

We use an input resolution of 128×128 for all experiments. The model is trained on images resized to this resolution to standardize input dimensions and maintain computational efficiency. For data augmentation, we apply random cropping, flipping, and color jittering during training to improve model generalization. The input data consists of both real and synthetic hand pose datasets, with the latter being uniformly distributed over 1520 poses and 216 viewpoints, ensuring comprehensive coverage of diverse hand configurations and viewpoints. The Adam optimizer is used for training with an initial learning rate of 10^{-3} , which is reduced $10 \times$ after the 30th epoch. We use a mini-batch size of 32, and the models are trained for a total of 38 epochs. The hyperparameters for our model follow the configuration specified in the baseline work [19], ensuring consistency and fairness in comparison.

Table 1 presents the results of our unit test. With 80% sparsity, the model achieves 39 fps on ResNet-18, delivering a speed-up of at least 2.1x over the dense convolution, whose frame rate remains constant. Our results demonstrate that sparse convolution effectively leverages sparse data derived from RGB images to achieve a 2 \times to 3 \times speed-up in frame rate. In contrast, dense convolution processes both sparse and dense inputs uniformly, showing no improvement in performance.

Table 1: Frame rate of sparse/dense convolution. FPS increases with higher input sparsity in sparse convolution.

| Architecture | 80% | 85% | 90% |
|------------------|-----------|-----------|-----------|
| ResNet-18 | 18 | 18 | 19 |
| Sparse ResNet-18 | 39 | 42 | 51 |
| ResNet-50 | 4 | 3 | 4 |
| Sparse ResNet-50 | 16 | 18 | 20 |

Table 4 illustrates our model’s ability to generalize on multimodal data. The results of applying Quantization-Aware Training (QAT) are shown in Table 4. Accuracy, measured on RGB images, remains comparable, with PA-MPJPE largely unchanged before and after QAT. Additionally, we train our model with multimodal data by transforming the FreiHAND dataset [25] into edge images and using our fusion module. Testing on edge images with and without QAT yields PA-MPJPE values of 12.3mm and 11.2mm, respectively, demonstrating the model’s robustness across different modalities.

5.2 Comparison with existing method

We utilize compound benchmark dataset. Rendered Hand Pose Dataset (RHD) [35] consists of 41,258 and 2,728 synthetic hand data for training and testing on hand pose estimation, respectively. And FreiHAND [25] contains 130,240 training images and 3,960 evaluation samples.

Table 2: Comparison of lightweight model speed, parameter and accuracy on the benchmark dataset [25] First and Second best results are represented as **RED** and **BLUE**.

| Method | FPS↑ | Params (M)↓ | PJ (mm)↓ |
|-------------------------|-----------|-------------|------------|
| METRO [36] | 2 | 102 | 6.7 |
| MobileHand [8] | 35 | 3.2 | 13 |
| SimpleHand [37] | 4 | 1.9 | 5.8 |
| FastMETRO [38] | 3 | 25 | 6.5 |
| MeshGraphomer [39] | 1 | 98 | 6.3 |
| MobRecon(baseline) [19] | 6 | 8.16 | 9.2 |
| Ours | 15 | 8.79 | 9.1 |

MobRecon [19] serves as our baseline because it represents one of the most efficient and lightweight architectures proposed in recent years at a top-tier conference, offering a strong balance between accuracy and computational cost. Its design makes it especially suitable for mobile and resource-constrained environments, which

aligns with the goals of our work. Table 2 presents a performance comparison between our proposed model and several state-of-the-art lightweight models. To highlight the practical efficiency of our model, we conducted a real-world evaluation on a Raspberry Pi 5 CPU.

Our model achieves a 3x improvement in inference speed while maintaining comparable accuracy on the compound benchmark dataset [25, 35]. While Guan *et al.* [8] remains the fastest, its model suffers a significant 42.8%. Our framework strikes a powerful balance between speed and accuracy. While SimpleHand [37] shows better accuracy, we achieve a substantial speed improvement, running at 15 FPS – 3.4 \times faster. Likewise, we outperform MobileHand [8] and MobRecon [19] in accuracy (PA-MPJPE) and speed(FPS).

Unlike other transformer-based methods like FastMETRO [36], MeshGraphomer [39], and SimpleHand [37], which struggle with real-time performance on edge devices, our model is highly efficient. While our model is not the most compact in terms of parameters, its architecture, which incorporates sparse convolution, is specifically optimized for efficiency. For all models, inference time was measured on a Raspberry Pi 5 CPU and represents the mean over 50 repetitions. The number of parameters is listed in millions (M), and Frames Per Second (FPS) is calculated as the inverse of the forward pass time. This edge device-based evaluation clearly demonstrates the effective acceleration and real-world applicability of our model.

Table 3 presents a controlled unit test with identical input shapes. The proposed SPLite module achieves faster inference (averaged over 50 repetitions) while significantly reducing both FLOPs and parameter counts. This is accomplished by convolving only a quarter of the feature channels, thereby reducing memory access costs, combined with parallel vertex sampling to accelerate traversal sampling process in spiral convolution [22]. Together, these design choices deliver a substantial frame rate improvement over the baseline [19].

Despite these optimizations, the overall parameter count of our end-to-end model remains substantial. This is because the primary operations of a typical encoding-decoding framework are in the 2D encoding stage. Since the ResNet-18 encoder is relatively large, the extensive reduction in our decoder’s parameters has a limited impact on the total model size.

Table 3: The controlled unit test of our SPLite module w.r.t. SpiralConv++; Inference time(IF) is tested on Rasberry Pi 5 CPU, which is the forward time in the network; [25].

| 3D decoding | Params(K) | FLOPs(M) ↓ | IF(ms) ↓ |
|-------------------|-----------|-------------|-------------|
| SpiralConv [22]++ | 21K | 31.87 | 0.78 |
| SPLite(ours) | 5K | 9.36 | 0.72 |

5.3 Qualitative Results

In Figure 2, we evaluate our model’s performance on challenging real-world images. The results highlight the limitations of the current fastest method by Lim *et al.* [8], which struggles to produce accurate 3D hand pose estimations on such real-world data. In contrast, our approach significantly outperforms by predicting 3D

Table 4: Accuracy on multimodality data after applying Quantization-Aware training

| precision (bits) | RGB PJ (mm)↓ | Edge PJ(mm)↓ | Model size(MB)↓ |
|------------------|--------------|--------------|-----------------|
| FP32(w/o QAT) | 9.0 | 11.2 | 71 |
| INT8(w/ QAT) | 9.1 | 12.3 | 18 |

hand poses that are much closer to the true hand configurations. The figure illustrates a side-by-side comparison: the input images, our model’s 3D joint estimations, 3D mesh reconstructions, and the outputs from MobileHand[8] and Mobrecon[19] for reference. Our method consistently generates more precise and anatomically plausible hand shapes and poses, demonstrating its robustness and superior generalization to wild, complex scenes.

In Figure 3, We processed real-world data using a proprietary algorithm to generate the edge images shown in the leftmost column. As illustrated in the figure, our method demonstrates superior accuracy in predicting the 3D hand joints and mesh. In contrast, existing approaches like MobileHand [8] and Mobrecon [19] appear to fail when presented with this different modality.

Table 5: Comparison of popular RGB-based real-world 3D hand datasets.

| Dataset | Size | Mesh | Multi-view | Multi-modality |
|--------------------|-------|------|------------|----------------|
| STB [40] | 36K | ✗ | ✗ | ✗ |
| EgoDexter [41] | 3K | ✗ | ✗ | ✗ |
| Dexter+Object [42] | 3K | ✗ | ✗ | ✗ |
| FreiHAND [25] | 134K | ✓ | ✗ | ✗ |
| YoutubeHand [43] | 47K | ✓ | ✗ | ✗ |
| HO3D [44] | 77K | ✓ | ✗ | ✗ |
| DexYCB [45] | 528K | ✓ | ✗ | ✗ |
| H3D [46] | 22K | ✓ | ✓(15) | ✗ |
| MHP [47] | 80K | ✓ | ✓(4) | ✗ |
| HOI4D [48] | 2.4M | ✓ | ✓ | ✗ |
| HOT3D [48] | 1.16M | ✓ | ✓(4) | ✗ |
| ours | 135K | ✓ | ✓(2) | ✓(3) |

In Table 5, we compare several prominent 3D hand pose datasets. HOI4D [31] comprises 2.4 million egocentric video frames and is widely used in robotic training; it includes 16 object categories collected from 4 participants. HOT3D [32], captured using Meta’s headset devices [49], comprises 1.16 million well-annotated frames from 19 participants, spanning 33 object categories. In contrast, our dataset contains 135 thousand video frames collected from 20 participants, each interacting with 5 unique object categories—resulting in a total of 100 categories (Figure 5). Since object manipulation patterns vary across individuals, our dataset offers greater diversity in interaction styles, providing a valuable resource for robotics training.

6 conclusion

In this work, we present a novel multimodal dataset and a light-weight deep learning framework designed to enhance hand-object

interaction understanding, specifically targeting robotics and VR applications. Our dataset with around object categories and action manipulations outperforms existing ones. The proposed framework, utilizing a sparse convolution approach on a ResNet-18 backbone and the innovative SPLite decoder, achieves a significant performance boost, and send a significant signal for edge-device deployment. Through quantization-aware training, we maintain high accuracy while reducing memory usage. On a Raspberry Pi 5, our method delivers nearly a 3x speed-up compared to traditional methods, making it an efficient solution for real-time inference on edge devices.

Future research can explore further optimizations for edge device deployment, including the integration of more advanced compression techniques and hardware accelerators to enhance inference speed and energy efficiency. Additionally, expanding the dataset to include more diverse scenarios and interactions can further improve the robustness and generalization of models. We also plan to investigate the application of this framework in other domains, such as AR and teleoperation, where real-time performance and low-latency interactions are critical.

Acknowledgments

This work is supported by the National Science and Technology Council (NSTC), R.O.C., under the projects “Advanced Technology of Intelligent Sensing Chip – Applied to Hand Pose Recognition System” and “NexIS: Next-Generation Intelligent Services through Self-Supervised and Trustworthy Learning Technologies.” We are also grateful for the support and guidance from Hsu Tzu Wei and Min Sun. Finally, we thank the anonymous reviewers for their valuable feedback.

References

- [1] Kevin Lin, Lijuan Wang, and Zicheng Liu. End-to-end human pose and mesh reconstruction with transformers, 2021.
- [2] Xingyu Liu, Pengfei Ren, Yuanyuan Gao, Jingyu Wang, Haifeng Sun, Qi Qi, Zirui Zhuang, and Jianxin Liao. Keypoint fusion for rgb-d based 3d hand pose estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 3756–3764, 2024.
- [3] Jia Guo, Jiankang Deng, Niannan Xue, and Stefanos Zafeiriou. Stacked dense u-nets with dual transformers for robust face alignment. *arXiv preprint arXiv:1812.01936*, 2018.
- [4] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.
- [5] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.
- [6] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices, 2017.
- [7] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1580–1589, 2020.
- [8] Guan Ming Lim, Prayook Jatesikat, and Wei Tech Ang. Mobilehand: Real-time 3d hand shape and pose estimation from color image. In *International conference on neural information processing*, pages 450–459. Springer, 2020.
- [9] Hongduk Choi, Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. Beyond static features for temporally consistent 3d human pose and shape from a video, 2021.
- [10] Jiefeng Li, Siyuan Bian, Ailing Zeng, Can Wang, Bo Pang, Wentao Liu, and Cewu Lu. Human pose regression with residual log-likelihood estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11025–11034, 2021.
- [11] Yang He and Lingxiao Xiao. Structured pruning for deep convolutional neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(5):2900–2919, 2023.
- [12] Guan Li, Junpeng Wang, Han-Wei Shen, Kaixin Chen, Guihua Shan, and Zhonghua Lu. Cnnpruner: Pruning convolutional neural networks with visual

- analytics. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1364–1373, 2020.
- [13] Peisong Wang, Qinghao Hu, Yifan Zhang, Chunjie Zhang, Yang Liu, and Jian Cheng. Two-step quantization for low-bit neural networks. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pages 4376–4384, 2018.
- [14] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3009–3018. IEEE, 2019.
- [15] Takashi Fukuda, Masayuki Suzuki, Gakuto Kurata, Samuel Thomas, Jia Cui, and Bhuvana Ramabhadran. Efficient knowledge distillation from an ensemble of teachers. In *Interspeech*, pages 3697–3701, 2017.
- [16] Nima Aghili and Eraldo Ribeiro. Combining weight pruning and knowledge distillation for cnn compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3191–3198, 2021.
- [17] Jierun Chen, Shiu-hong Kao, Hao He, Weipeng Zhuo, Song Wen, Chul-Ho Lee, and S. H. Gary Chan. Run, don't walk: Chasing higher flops for faster neural networks, 2023.
- [18] Sen Zhang, Fusheng Zha, Xiangji Wang, Mantian Li, Wei Guo, Pengfei Wang, Xiaolin Li, and Lining Sun. High-efficiency sparse convolution operator for event-based cameras. *Frontiers in Neurorobotics*, 19:1537673, 2025.
- [19] Xingyu Chen, Yufeng Liu, Yajiao Dong, Xiong Zhang, Chongyang Ma, Yanmin Xiong, Yuan Zhang, and Xiaoyan Guo. Mobrecon: Mobile-friendly hand mesh reconstruction from monocular image, 2022.
- [20] Yuxin Chen, Ziqi Zhang, Chunfeng Yuan, Bing Li, Ying Deng, and Weiming Hu. Channel-wise topology refinement graph convolution for skeleton-based action recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13359–13368, 2021.
- [21] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.
- [22] Shunwang Gong, Lei Chen, Michael Bronstein, and Stefanos Zafeiriou. Spiralnet++: A fast and highly efficient mesh convolution operator, 2019.
- [23] Giorgos Bouritsas, Sergiy Bokhnyak, Stylianos Ploumpis, Michael Bronstein, and Stefanos Zafeiriou. Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7213–7222, 2019.
- [24] Saleh Ashkboos, Bram Verhoeft, Torsten Hoefer, Evangelos Eleftheriou, and Martino Dazzi. EfqaT: An efficient framework for quantization-aware training, 2024.
- [25] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images, 2019.
- [26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [27] Christopher Choy, Jaesik Park, and Vladlen Koltun. Minkowski engine: Sparse convolutional neural networks. <https://github.com/NVIDIA/MinkowskiEngine>, 2019. Accessed: 2025-08-08.
- [28] Haotian Tang, Shang Yang, Zhijian Liu, Ke Hong, Zhongming Yu, Xiuyu Li, Guohao Dai, Yu Wang, and Song Han. Torchsparsen++: Efficient training and inference framework for sparse convolution on gpus. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 225–239, 2023.
- [29] Ting-Ying Lin, Lin-Yung Hsieh, Fu-En Wang, Wen-Shen Wuen, and Min Sun. Sparse and privacy-enhanced representation for human pose estimation. In *BMVC*, 2023.
- [30] Shreyas Hampali, Sayan Deb Sarkar, and Vincent Lepetit. Ho-3d_v3: Improving the accuracy of hand-object annotations of the ho-3d dataset. *arXiv preprint arXiv:2107.00887*, 2021.
- [31] 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, 2024.
- [32] Prithviraj Banerjee, Sindi Shkodrani, Pierre Borschow, Shreyas Hampali, Shangchen Han, Fan Zhang, Lingguang Zhang, Jade Fountain, Edward Miller, Selen Basol, Richard Newcombe, Robert Wang, Jakob Julian Engel, and Tomas Hodan. Hot3d: Hand and object tracking in 3d from egocentric multi-view videos, 2025.
- [33] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(6):679–698*, 1986.
- [34] Irwin Sobel and Gary Feldman. An isotropic 3×3 image gradient operator. In *Stanford Artificial Intelligence Project (SAIL)*, 1968. Presented at the Stanford Artificial Intelligence Laboratory (SAIL).
- [35] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single rgb images, 2017.
- [36] Kevin Lin, Lijuan Wang, and Zicheng Liu. End-to-end human pose and mesh reconstruction with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1954–1963, 2021.
- [37] Zhishan Zhou, Shihao Zhou, Zhi Lv, Minqiang Zou, Yao Tang, and Jiajun Liang. A simple baseline for efficient hand mesh reconstruction, 2024.
- [38] Junhyeong Cho, Kim Youwang, and Tae-Hyun Oh. Cross-attention of disentangled modalities for 3d human mesh recovery with transformers. In *European Conference on Computer Vision*, pages 342–359. Springer, 2022.
- [39] Kevin Lin, Lijuan Wang, and Zicheng Liu. Mesh graphomer, 2021.
- [40] Jiawei Zhang, Jianbo Jiao, Mingliang Chen, Liangqiong Qu, Xiaobin Xu, and Qingxiang Yang. 3d hand pose tracking and estimation using stereo matching. *arXiv preprint arXiv:1610.07214*, 2016.
- [41] Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. Real-time hand tracking under occlusion from an egocentric rgb-d sensor. In *Proceedings of the IEEE international conference on computer vision*, pages 1154–1163, 2017.
- [42] Srinath Sridhar, Franziska Mueller, Michael Zollhöfer, Dan Casas, Antti Oulasvirta, and Christian Theobalt. Real-time joint tracking of a hand manipulating an object from rgb-d input. In *European conference on computer vision*, pages 294–310. Springer, 2016.
- [43] Dominik Kulon, Riza Alp Guler, Iasonas Kokkinos, Michael M Bronstein, and Stefanos Zafeiriou. Weakly-supervised mesh-convolutional hand reconstruction in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4990–5000, 2020.
- [44] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Honnotate: A method for 3d annotation of hand and object poses. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3196–3206, 2020.
- [45] Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, et al. Dexycb: A benchmark for capturing hand grasping of objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9044–9053, 2021.
- [46] Linlin Yang, Shile Li, Dongheui Lee, and Angela Yao. Aligning latent spaces for 3d hand pose estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2335–2343, 2019.
- [47] Francisco Gomez-Donoso, Sergio Orts-Escolano, and Miguel Cazorla. Large-scale multiview 3d hand pose dataset. *Image and Vision Computing*, 81:25–33, 2019.
- [48] Adnane Boukhayma, Rodrigo de Bem, and Philip H. S. Torr. 3d hand shape and pose from images in the wild, 2019.
- [49] Jakob Engel, Kiran Somasundaram, Michael Goesele, Albert Sun, Alexander Gamino, Andrew Turner, Arjung Talatoff, Arnie Yuan, Bilal Souti, Brighid Mereditth, Cheng Peng, Chris Sweeney, Cole Wilson, Dan Barnes, Daniel DeTone, David Caruso, Derek Valleroy, Dinesh Ginjupalli, Duncan Frost, Edward Miller, Elias Mueggler, Evgeniy Oleinik, Fan Zhang, Guruprasad Somasundaram, Gustavo Solaira, Harry Lanaras, Henry Howard-Jenkins, Huixuan Tang, Hyo Jin Kim, Jaime Rivera, Ji Luo, Jing Dong, Julian Straub, Kevin Bailey, Kevin Eckenhoff, Lingni Ma, Luis Pesqueira, Mark Schwesinger, Maurizio Monge, Nan Yang, Nick Charron, Nikhil Raina, Omkar Parkhi, Peter Borschow, Pierre Moulon, Prince Gupta, Raul Mur-Artal, Robbie Pennington, Sachin Kulkarni, Sagar Miglani, Santosh Gondi, Saransh Solanki, Sean Diener, Shangyi Cheng, Simon Green, Steve Saarinen, Suvam Patra, Tassos Mourikis, Thomas Whelan, Tripti Singh, Vasileios Balntas, Vijay Baiyya, Wilson Dreeves, Xiaqing Pan, Yang Lou, Yipu Zhao, Yusuf Mansour, Yuyang Zou, Zhaoyang Lv, Zijian Wang, Mingfei Yan, Carl Ren, Renzo De Nardi, and Richard Newcombe. Project aria: A new tool for egocentric multi-modal ai research, 2023.

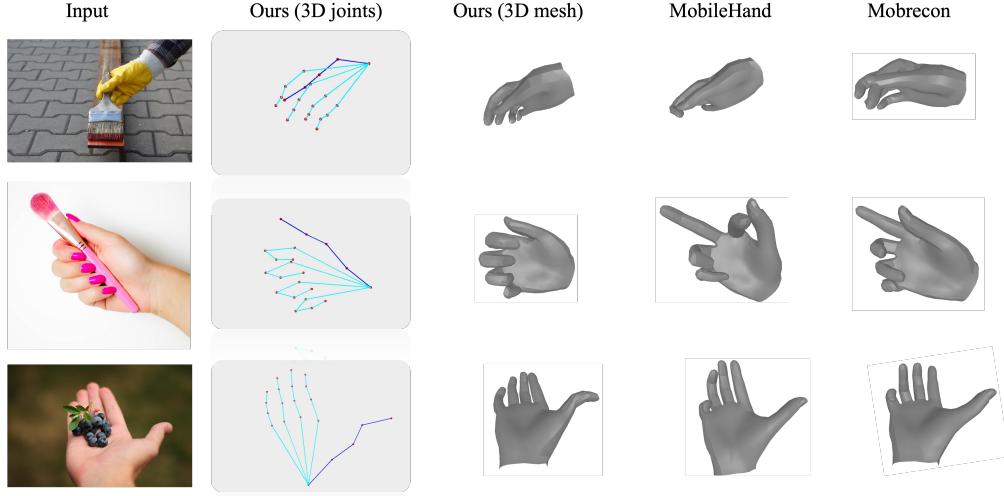


Figure 2: Qualitative comparison on in-the-wild samples. Each row shows an input RGB image (left), followed by our predicted 3D joints, our reconstructed 3D mesh, and results from MobileHand[8] and Mobrecon[19]. Our method produces more accurate and natural hand poses, preserving fine articulation and finger alignment, while competing methods exhibit noticeable distortions or misalignments, particularly in challenging poses and occlusions.

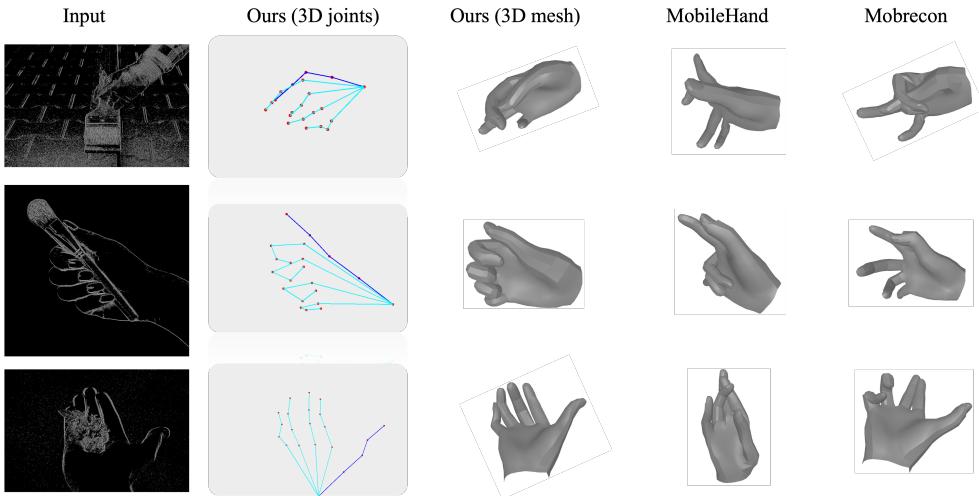


Figure 3: Qualitative comparison on in-the-wild samples. Each row shows an input edge image (left). Our method produces accurate and natural hand poses, while competing methods [8] [19] demonstrate unreasonable distortions and fail to produce a coherent hand structure on this modality.

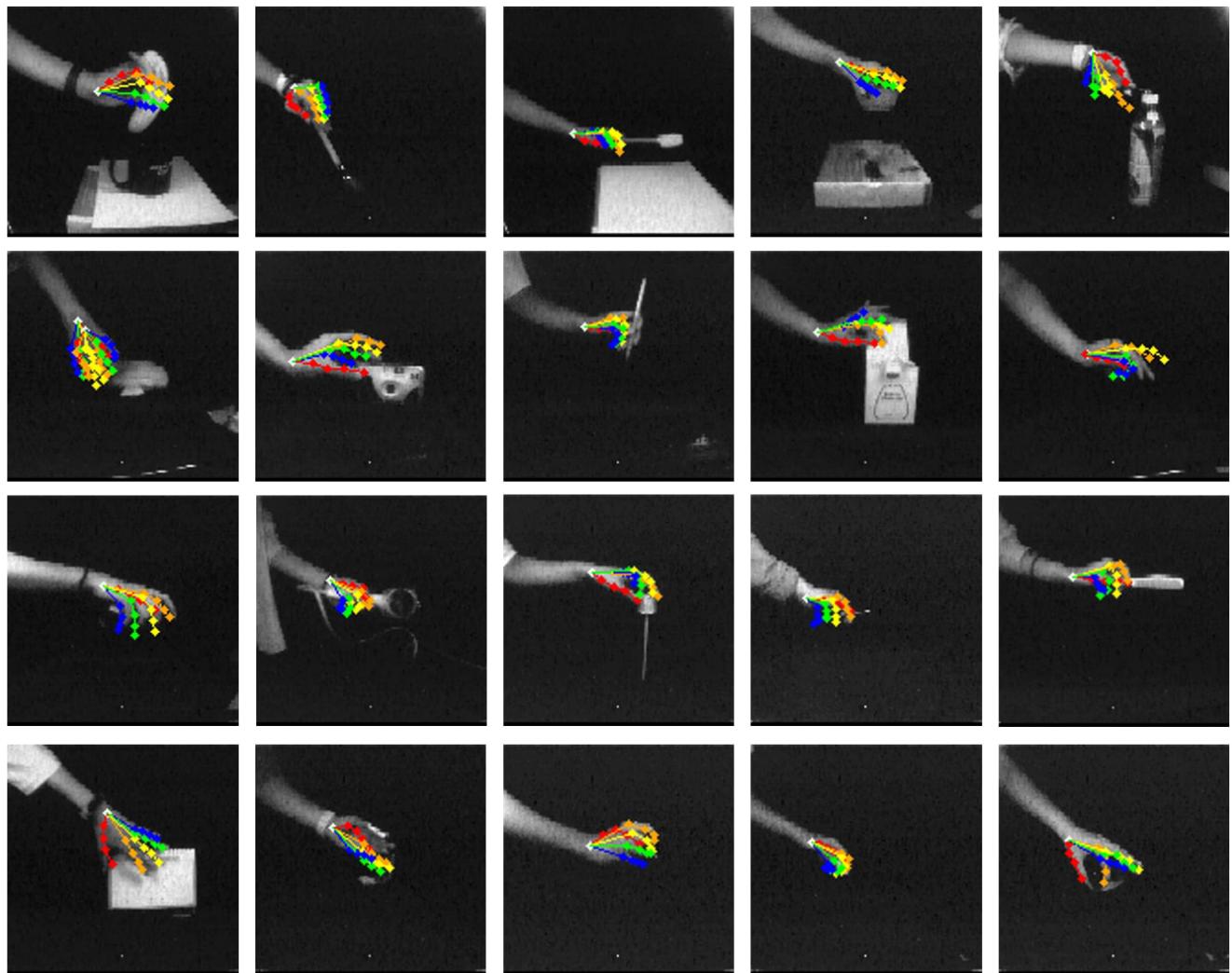


Figure 4: The dataset is captured in real-world scenarios using a proprietary sensor, yielding 128x128 resolution images synchronized with two Intel RealSense cameras from different perspectives. The dataset includes four distinct scenarios, as shown in Figure 5.100 genres of object categories and corresponding action manipulation respectively. The ground-truth keypoints and meshes were labeled semi-manually.

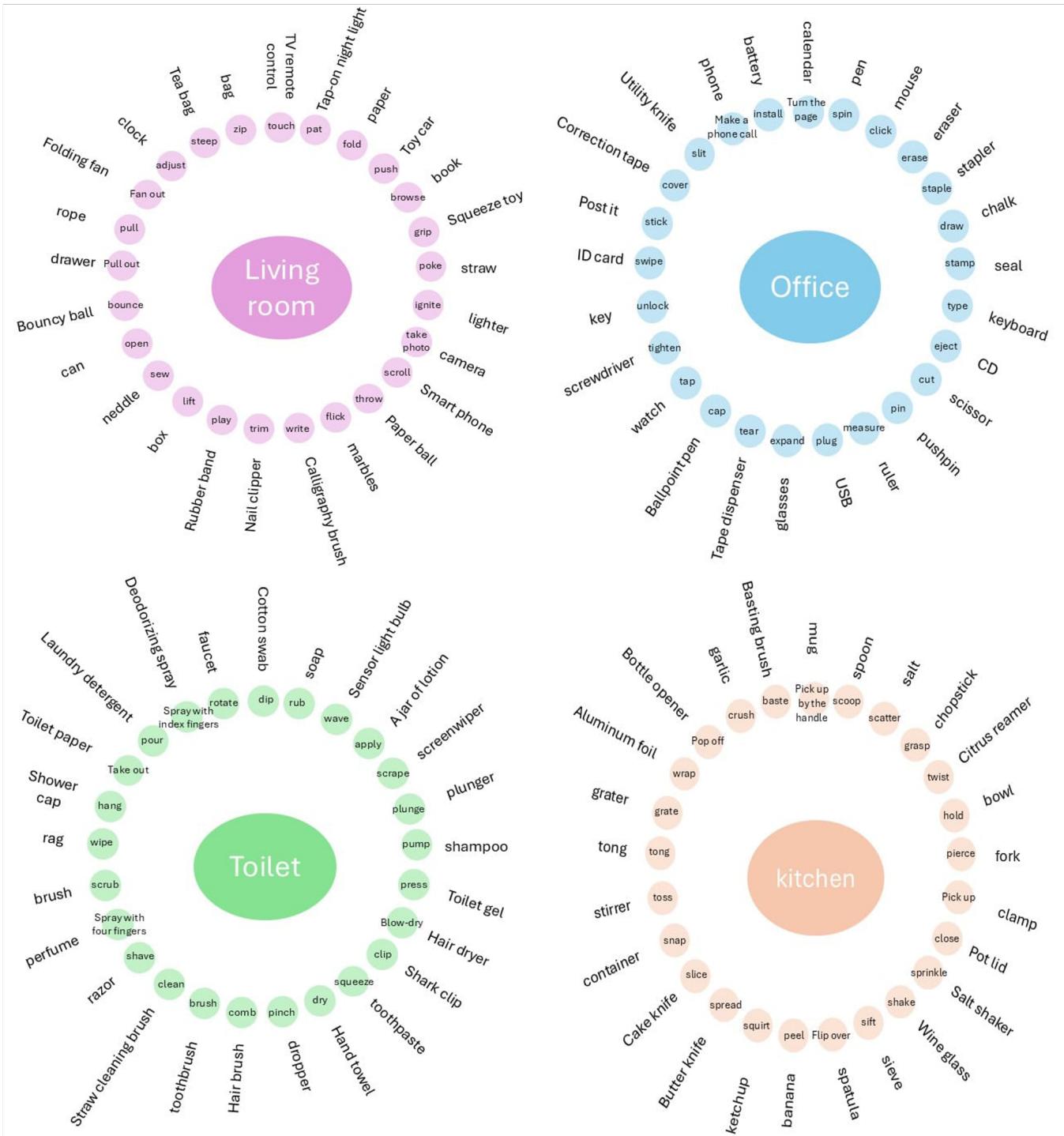


Figure 5: Illustration of our dataset, which includes object and motion categories—each with 25 unique objects and corresponding manipulation actions. We designed four common daily scenarios in the dataset to aid in robot model training. Images were captured using two Intel RealSense cameras and one proprietary camera. Ground-truth keypoints and meshes were semi-manually labeled, as detailed in the Appendix.