ARCTIC: A Dataset for Dexterous Bimanual Hand-Object Manipulation

Zicong Fan[1,3] Omid Taheri[3] Dimitrios Tzionas[2] Muhammed Kocabas[1,3]
Manuel Kaufmann[1] Michael J. Black[3] Otmar Hilliges[1]
[1]ETH Zürich, Switzerland [2]University of Amsterdam [3]Max Planck Institute for Intelligent Systems, Tübingen, Germany

1. Dataset Details

Objects in ARCTIC: Figure 1 shows all 11 articulated objects in our dataset. Objects in ARCTIC consists of two rigid parts that rotate about an axis. Each dash line in the figure shows the articulation axis.

Marker sets: Figure 2 shows marker sets for an object, the full human body, two hands, and the egocentric camera along with the marker size. Markers in this visualization are shown to scale. The marker locations for all objects and all subjects can be found in the data release.

Dataset statistics: Table 1 shows the number of images and the number of sequences per subject for our dataset. The average sequence length is 698 frames (view-agnostic), corresponding to 23.3 seconds. In total, we have 2.1M images and there are more than 200k images for most subjects. Table 2 shows the number of images per object. All objects have more than 170k images. To encourage different modes of interaction, we capture different intents for each object: "use" and "grasp". Although both are for dexterous manipulation, in the "use" sequences, the subjects are allowed to articulate the object but not in the "grasp". Since we focus on studying articulation, we capture more "use" sequences.

Protocol splits: Table 3 shows the number of images and subjects in the allocentric and the egocentric settings. Both settings use the same subject split – 8 subjects for training, 1 for validation and 1 for testing. The allocentric setting uses images from the 8 allocentric static views for training, validation, and testing. The egocentric setting, in the training split, we allow models to use images from all 9 views for additional supervision; During inference, however, models are evaluated with only egocentric images.

Depth images: Since we perform full-body capture, we can render depth images with full-body interaction. Since most existing articulated object datasets contain neither two-hands nor human bodies [15, 16, 23], and having a human in the scene is a realistic setting, we believe that ARCTIC brings additional challenges of heavy occlusion and dynamic manipulation to the depth community. Figure 3 shows examples of the depth images and the corresponding RGB images. The depth can be rendered with any synthetic sensor noise model (e.g., Kinect, right).

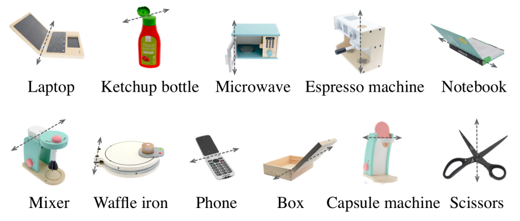


Figure 1. **ARCTIC objects**. Each line shows the articulation axis.

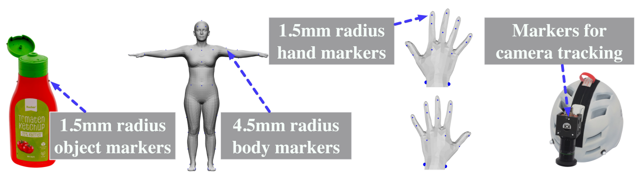

1.5mm radius hand markers
Markers for camera tracking
1.5mm radius object markers
4.5mm radius body markers

Figure 2. **Markers for motion capture (MoCap)**. We put 1.5mm radius markers on objects, hands and the egocentric camera. For the body, we use 4.5mm radius markers. The markers are shown here to scale. Best viewed in color and zoomed-in.

Subjects	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	Total
# Images	209k	224k	220k	212k	227k	228k	191k	280k	208k	135k	2.1M
# Seqs	34	37	38	31	34	36	29	42	37	21	339

Table 1. **Number of images and sequences for each subject**. The average sequence length is 698 frames (view-agnostic), corresponding to 23.3 seconds.

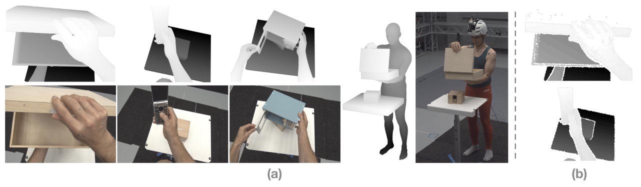


Figure 3. **Rendered depth images of human-object interaction.** (a) Rendered depth images of our 3D data, and the corresponding RGB images, and (b) depth images with synthetic kinect noise. Best viewed zoomed-in. For better depth-map visualization we do not render the floor here.

Objects	Notebook	Box	Espresso machine	Waffle iron	Laptop	Phone	Capsule machine	Mixer	Ketchup bottle	Scissors	Microwave	Total
Use	163k	152k	141k	166k	171k	159k	159k	156k	138k	128k	144k	1.7M
Grasp	27k	36k	46k	41k	40k	49k	37k	47k	51k	44k	43k	0.4M
Total	190k	187k	187k	207k	211k	208k	196k	202k	189k	172k	186k	2.1M

Table 2. **Number of images for each object in ARCTIC**. In ARCTIC, we focus on studying hand interaction with object articulation. Therefore, we capture more "use" sequences in which subjects can articulate the object. To encourage different modes of interaction, we also ask subjects to "grasp" the objects without articulating the objects. Since we focus on object articulation, we capture more data for "use" than in "grasp".

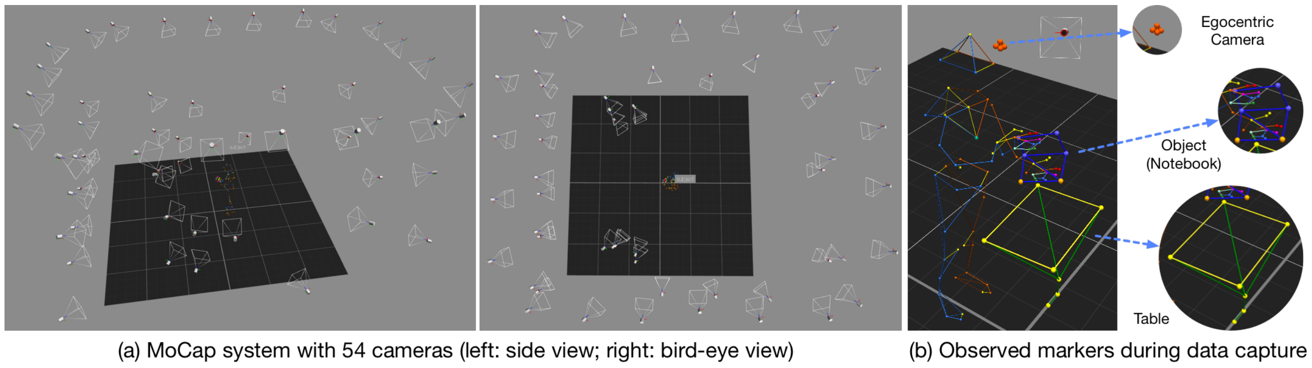

Egocentric Camera
Object (Notebook)
Table
(a) MoCap system with 54 cameras (left: side view; right: bird-eye view)
(b) Observed markers during data capture

Figure 4. **Our capture system with** 54 **high-resolution MoCap cameras**. (a) MoCap system in a side view and a bird-eye view, illustrating the 54 MoCap cameras used to eliminate occlusion during the capture. (b) Observed markers for a captured frame, showing markers tracking the full human body with hands, the object (notebook in this case), the egocentric camera, and props such as the table. Best viewed in color and zoomed in.

Splits	# Train Images	# Val Images	# Test Images
allo	1.5M	202k	195k
ego	1.7M	25k	24k

Table 3. **Number of images for each protocol**.

2. Data Capture Details

2.1. MoCap System with 54 Cameras

Figure 4 illustrates our MoCap system. When capturing quality hand-object interaction data, the key is to eliminate occlusion in hand self-occlusion, hand-hand occlusion, and hand-object occlusion settings. To minimize these sources of occlusion, we use 54 high-resolution MoCap cameras during our capture. The camera positions and orientations are shown in Fig. 4a in a side view and a bird-eye view. We also show the markers tracked by our system in Fig. 4b. The system tracks markers on the egocentric camera, the human subject (full body with hands), the articulated object (notebook in this case), and props such as the table.

2.2. Creating Personalized Template

To create a personalized full body-and-hand template mesh for each subject, we obtain 3D scans of the subjects in varying poses using a 3dMD scanner [22]. We then register the SMPL-X model to the scans to obtain aligned meshes. The registered SMPL-X meshes are unposed to a canonical T-Pose. We perform a SMPL-X model-based fitting to the unposed meshes using vertex-to-vertex distances with the SMPL-X vertex correspondences. Fitting with multiple T-Posed meshes allows us to filter out potential noise, and to capture the occluded regions of the body and hands, resulting in a reliable personalized SMPL-X template for each subject.

2.3. Estimating Rotation Axis and Articulated Pose

To solve for the rotation axis of each articulated object, we attach markers on each rigid part of each object and capture a calibration MoCap sequence by articulating its two parts. Since the two parts rotate about an axis, the trajectory of each marker follows a circle on a 2D plane. We solve for the center of each circle using least-squares, and fit a 3D line through the centers to obtain an initial rotation axis estimate. We then refine the rotation axis estimate by minimizing a cost function.

Formally, let $X_t^i \in \mathbb{R}^3$ denote the 3D position of a marker i at time step t placed on the "top" part of the object (*e.g.*, the lid of the ketchup bottle); $Y_t^j \in \mathbb{R}^3$ denotes a marker j on the "bottom" part of the object (*e.g.*, the main body of the ketchup bottle) at time t. We pick a frame t_0 corresponding to a pre-defined rest pose for each object (for example, a frame in which the lid of the ketchup bottle is closed). We then transform the MoCap sequence $\{X_t^i\}_{i,t}$ and $\{Y_t^j\}_{j,t}$ into a canonical sequence $\{\bar{X}_t^i\}_{i,t}$ and $\{\bar{Y}_t^j\}_{j,t}$

via a rigid transformation (R, T) such that $\bar{Y}_t^j = Y_{t_0}^j$ for all t and j. After the canonicalization, the markers on the bottom part of the object are stationary across the entire sequence, and the top markers rotate around an axis. Further, the trajectory of each marker is a circle on a 2D plane. We fit a circle to the trajectory $\{\bar{X}_t^i\}_{t=1,\cdots,N}$ of an arbitrary marker i using least-squares, and convert the 2D circle center to the 3D space (x_i, y_i, z_i). We then fit a 3D line (v, v_0) to the center of each circle $\{(x_i, y_i, z_i)\}_i$ in 3D, where $v \in \mathbb{R}^3$ is a unit directional vector for the line and $v_0 \in \mathbb{R}^3$ is an arbitrary anchor point that the line crosses. Since fitting a 3D line to 3D centers can be imprecise, we refine the rotation axis further by minimizing the following cost function

$$(v^*, v_0^*, \omega_{t=1,\cdots,N}^*) = \underset{v, v_0, \omega_{t=1,\cdots,N}}{\arg\min} \sum_i ||\bar{X}_t^i - f(\bar{X}_{t_0}^i | v, v_0, \omega_t)||_2^2 \tag{1}$$

where N is the number of frames in a sequence; $\omega_{t=1,\cdots,N}^*$ are the articulation angles in radians for all the frames relative to the rest pose. The quantities v^*, v_0^* are the refined rotation axis for the object. The function $f(\bar{X}_{t_0}^i | v, v_0, \omega_t)$ rotates $\{\bar{X}_t^i\}$ about the estimated axis (v^*, v_0^*) by ω_t, the amount of articulation. This ensures that the estimated rotation axis is consistent with the marker trajectory in the MoCap data.

To define the articulated object pose, we need to estimate the 1D articulation angle, and its 6D rigid pose. To compute the former, after the rotation axis (v^*, v_0^*) is estimated, for an actual MoCap sequence, the articulation angles are obtained by performing a 2D projection of the 3D marker positions along the rotation axis. Since the 2D projection lies on a circle, the articulation angle can be estimated arithmetically. The articulation angle is measured relative to each object's rest pose defined in t_0 during the rotation axis estimation step. We take the median of the articulation angles estimated from all markers at a time step as our ground-truth articulation angle. Finally, to define the articulated object pose, we also need the 6D object pose for its orientation and translation. To solve for the 6D pose (R_t, T_t) for a frame t, we compute the rigid transformation from $\bar{Y}_{t_0}$ to Y_t. In other words, we compute the 6D pose using the base marker according to its correspondence from the canonical space to the MoCap space.

2.4. Computing Hand-Object Binary Contact

We consider the two hands and the two parts of each articulated object as four watertight meshes for computing ground-truth binary contact labels. Given one mesh from the hands and one mesh from the object parts, we follow GRAB [21] to compute vertex-level contact. The main idea in GRAB is to label vertices on a mesh as in contact with another mesh based on two cases: "contact under-shooting" and "contact over-shooting". When vertices on a mesh are not inside another mesh, it is considered "under-shooting", and geometric proximity is used to label contact. When there is interpenetration between two meshes, for example, the thumb goes through a thin structure, the vertices of the thumb that "over-shoot" the thin structure are labeled as in contact as well as the vertices that are inside the structure. For more details, we refer readers to [21].

Nr.	Module	Details
	Hand Branch	
1	pool	AvgPool2d(output_size=1)
2	cam_init	Linear(in_dim=2048, out_dim=512, bias=True)
3	cam_init	ReLU()
4	cam_init	Linear(in_dim=512, out_dim=512, bias=True)
5	cam_init	ReLU()
6	cam_init	Linear(in_dim=512, out_dim=3, bias=True)
7	refine.fwd	Concat(["feat", "hand_pose", "cam", "shape"])
8	refine.fwd	Linear(in_dim=2157, out_dim=1024, bias=True)
9	refine.fwd	ReLU()
10	refine.fwd	Dropout(p=0.5)
11	refine.fwd	Linear(in_dim=1024, out_dim=1024, bias=True)
12	refine.fwd	ReLU()
13	refine.fwd	Dropout(p=0.5)
14	refine.decode.pose_6d	Linear(in_dim=1024, out_dim=96, bias=True)
15	refine.decode.shape	Linear(in_dim=1024, out_dim=10, bias=True)
16	refine.decode.cam	Linear(in_dim=1024, out_dim=3, bias=True)
	Object Branch	
1	pool	AvgPool2d(output_size=1)
2	cam_init	Linear(in_dim=2048, out_dim=512, bias=True)
3	cam_init	ReLU()
4	cam_init	Linear(in_dim=512, out_dim=512, bias=True)
5	cam_init	ReLU()
6	cam_init	Linear(in_dim=512, out_dim=3, bias=True)
7	refine.fwd	Concat(["feat", "rot", "cam", "arti"])
8	refine.fwd	Linear(in_dim=2055, out_dim=1024, bias=True)
9	refine.fwd	ReLU()
10	refine.fwd	Dropout(p=0.5)
11	refine.fwd	Linear(in_dim=1024, out_dim=1024, bias=True)
12	refine.fwd	ReLU()
13	refine.fwd	Dropout(p=0.5)
14	refine.decode.rot	Linear(in_dim=1024, out_dim=3, bias=True)
15	refine.decode.cam	Linear(in_dim=1024, out_dim=3, bias=True)
16	refine.decode.arti	Linear(in_dim=1024, out_dim=1, bias=True)

Table 4. **Details of the decoder in ArcticNet-SF.**

3. Model Details and Results

General implementation details: For all experiments, we use a ResNet-50 [19] backbone pre-trained on ImageNet [2]. The models are trained with the Adam optimizer [11] using a learning rate of $1e^{-5}$. For visibility, we crop each image around a square region centered around the object and resize the image to 224×224. Data augmentation is applied to the input image: rotation ($\pm 30°$), scaling ($\pm 25\%$), and color jittering ($\pm 40\%$).

3.1. ArcticNet

ArcticNet-SF Architecture: We show the details of ArcticNet-SF in Table 4. ArcticNet-SF uses an encoder-decoder architecture. Given an input image, we use average pooling to obtain a single image feature vector with dimension 2048 from the backbone. The image feature vector is used to predict the initial camera parameters (in "cam_init"). Following [10], we use an iterative refinement scheme to predict parameters of the hands and the objects. First, we initialize all parameters to zero except for the camera parameters, which we initialize with the prediction in "cam_init". For the hand branch, we concatenate the image feature vector, the initial hand poses, the hand shape vector into a single vector, and the predicted camera parameters for refinement (Line 7-16). Within the refinement step, we first predict a latent vector using an MLP (Line 7-13). The latent vector is then being decoded via different heads to the residuals for MANO joint angles, shape and camera parameters (Line 14-16). The decoded residuals are added to the current estimates of the parameters respectively and will be used as inputs for the next refinement step (Line 7-16 again). We have two iterations for the refinement. The object branch has a similar refinement scheme, but instead it predicts the object rotation, camera parameters, and object articulation. Following [13, 14], we use the 6D rotation representation in [27] for MANO joint angles. Following [1, 10, 13, 18, 25], for the predicted weak perspective camera parameters, we use a fixed focal length of 1000.0 and convert them to translations for each entity in the scene.

ArcticNet-LSTM Architecture: The LSTM model takes in a moving window of images and estimates 3D meshes for each frame. We use the same structure as ArcticNet-SF except that the image features within each window are passed through an LSTM network before being decoded. We use a bidirectional LSTM with two hidden layers, hidden dimension of 1024, and a window size of 11 based on validation.

Training losses: For each frame, our loss $\mathcal{L}$ is defined as the sum of the left hand, right hand, object, and interaction losses: $\mathcal{L} = \mathcal{L}_l + \mathcal{L}_r + \mathcal{L}_o + \mathcal{L}_{int}$. In particular, the hand losses are defined as

$$\mathcal{L}_h = \lambda^h_{3D}\mathcal{L}^h_{3D} + \lambda^h_{2D}\mathcal{L}^h_{2D} + \lambda^h_{\Theta}\mathcal{L}^h_{\Theta} + \lambda^h_{T}\mathcal{L}^h_{T}, \quad (2)$$

where $h = \{l, r\}$ denotes the handedness. We fully supervise the 3D joints (after subtracting the roots), the 2D re-projection of the predicted 3D joints, the MANO pose and shape parameters and the weak-perspective camera parameters. Similarly, we pre-define 3D landmarks for objects using farthest point sampling [4, 5] on the object mesh. Using these landmarks, we formulate the object losses as

$$\mathcal{L}_o = \lambda^o_{3D}\mathcal{L}^o_{3D} + \lambda^o_{2D}\mathcal{L}^o_{2D} + \lambda_{\omega}\mathcal{L}_{\omega} + \lambda_R\mathcal{L}_R + \lambda^o_T\mathcal{L}^o_T, \quad (3)$$

where $\mathcal{L}_{\omega}$, $\mathcal{L}_R$ and $\mathcal{L}^o_T$ supervise the articulation angle in radians, the global orientation and the weak perspective camera parameters. For the interaction loss $\mathcal{L}_{int}$, we use the contact deviation (CDev) metric (see main paper) as a loss term to improve hand-object contact. We apply this loss between the left-hand/object, and right-hand/object. The loss $\mathcal{L}_{int}$ is a sum of the two. All losses above use the MSE criterion. All λ variables are hyper-parameters and are set empirically based on validation performance. In particular, we set all λs to 1.0 except $\lambda^*_{3D} = 5.0$, $\lambda^h_{2D} = 5.0$, $\lambda^h_{\Theta} = 10.0$, $\lambda^h_{\beta} = 0.001$ where $*$ denote a hand or an object.

Training details: We train with a batch size of 64. For the allocentric setting, we train single-frame models for 20 epochs. Since training temporal model is computation intensive, following VIBE [12], we dump image features of pre-trained single-frame models to disk then train the LSTM models directly on the image features for 10 epochs. For the egocentric setting, since a model has access to both allocentric and egocentric images during training, to speed up training, we finetune pre-trained allocentric models on egocentric training images (1 camera) for 50 epochs.

Camera model: Following previous work on body and hand surface reconstruction [1, 10, 13, 14, 18, 25], to estimate the translation of hands and objects (T_l, T_r, T_o), we predict weak-perspective camera parameters (s, t_x, t_y) for each entity in the scene. The camera parameters consist of the scale $s \in \mathbb{R}$ and translation $(t_x, t_y) \in \mathbb{R}^2$ in pixel space and the translation can be recovered from (s, t_x, t_y) [13, 14] via:

$$T = (t_x, t_y, \frac{2f}{ws}) \in \mathbb{R}^3. \quad (4)$$

The terms w and f are the patch size and the focal length. We do this for each (T_l, T_r, T_o).

Qualitative Results: Figure 5 shows the predictions of ArcticNet-SF and ArcticNet-LSTM on the test set. As shown in the quantitative results in the main manuscript, the ArcticNet-LSTM model has lower errors overall for its prediction and it has better contact. This is consistent with the observations in the qualitative examples here. We hypothesize that this is because the LSTM allows the network to jointly reason between the motions of hands and objects.

3.2. InterField

InterField-SF Architecture: Table 5 details our InterField model. As an example, we illustrate how the right hand interaction field is predicted. The left hand, and the object are predicted in a similar way. In particular, from an input image, we obtain a 2048-dimensional image feature vector from the image backbone. The vector is passed through an MLP and is projected to lower dimension for computational efficiency (Line 1-4). We use subsampled vertices of the hand, and concatenate the 3D location of each vertex of the subsampled hand in the canonical pose with the 512-dimensional image feature vector, resulting a point cloud with 515 dimensions. The point cloud is passed through a PointNet backbone to obtain a latent point cloud with 512 dimensions (Line 5-11). Within the PointNet backbone, the 515-dimensional input point cloud is passed through a sequence of layers to produce lower level point features (Line

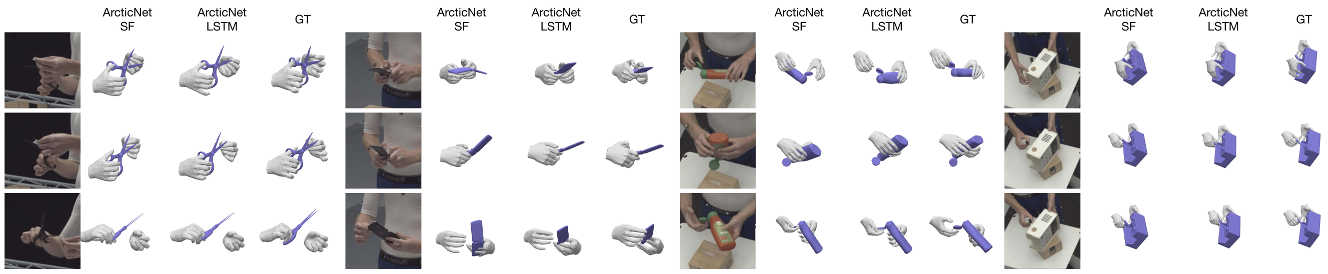


Figure 5. **Qualitative results of ArcticNet-SF and ArcticNet-LSTM**. Best viewed in color and zoomed in.

Nr.	Module	Details
1	img_feat.down	Linear(in_dim=2048, out_dim=512, bias=True)
2	img_feat.down	ReLU()
3	img_feat.down	Linear(in_dim=512, out_dim=512, bias=True)
4	img_feat.down	ReLU()
5	pointnet.shadow	Linear(in_dim=515, out_dim=512, bias=True)
6	pointnet.shadow	BatchNorm1d(512, affine=True)
7	pointnet.deep	Linear(in_dim=515, out_dim=512, bias=True)
8	pointnet.deep	BatchNorm1d(512, affine=True)
9	pointnet.deep	ReLU()
10	pointnet.deep	Linear(in_dim=515, out_dim=512, bias=True)
11	pointnet.deep	BatchNorm1d(512, affine=True)
12	regressor	Linear(in_dim=1024, out_dim=512, bias=True)
13	regressor	BatchNorm1d(512, affine=True)
14	regressor	ReLU()
15	regressor	Linear(in_dim=512, out_dim=128, bias=True)
16	regressor	BatchNorm1d(128, affine=True)
17	regressor	ReLU()
18	regressor	Linear(in_dim=128, out_dim=1, bias=True)
19	upsample	Linear(in_dim=195, out_dim=778, bias=True)

Table 5. **Details of InterField-SF architecture.**

5-6). The point features are further processed through Line 7-11. We then concatenate the point cloud from the shallow layers (output of Line 6) and the deeper layers (output of Line 11) along the feature dimension, resulting in a point cloud whose individual points are in 1024-dimensional. A regressor maps each point (1024-dimensional) to a single scalar for distance prediction (Line 12-18). Finally, we upsample the subsampled distances to the full hand mesh (Line 19). We predict the interaction field of the left hand and the object in the same way. All entities shared the same image and PointNet backbones.

InterField-LSTM Architecture: The LSTM model takes in images from a window and estimates the interaction field for each frame. In particular, we use the same architecture as in ArcticNet-SF except that we pass the image features in a window to an LSTM network before regressing the distances. We use a bidirectional LSTM with two hidden layers, hidden dimension of 1024 and a window size of 11 based on validation performance.

Training details: For each frame, the network outputs are $\hat{\boldsymbol{F}}^{l\to o}$, $\hat{\boldsymbol{F}}^{r\to o}$, $\hat{\boldsymbol{F}}^{o\to l}$, and $\hat{\boldsymbol{F}}^{o\to r}$. To supervise training, we extract the ground-truth interaction fields for each frame from ARCTIC and formulate an L1 loss $\mathcal{L} = \mathcal{L}_F(l,o) + \mathcal{L}_F(r,o) + \mathcal{L}_F(o,l) + \mathcal{L}_F(o,r)$ where $\mathcal{L}_F(a,b) = ||\boldsymbol{F}^{a\to b} - \hat{\boldsymbol{F}}^{a\to b}||_1$ for entities a and b. For tractability and focus on close interaction, we threshold the interaction field distances at 10cm for training and evaluation.

We train with a batch size of 64 for single-frame models and 32 for LSTM models. For the allocentric setting, we train single-frame models for 20 epochs. Since training temporal model has high computational requirements, following [12], we dump image features of pre-trained single-frame networks to disk and train the LSTM models on the image features for 6 epochs. For the egocentric setting, a model has access to both allocentric and egocentric images in the training set. To speed up training, we finetune pre-trained allocentric models on egocentric images (1 camera) for 100 epochs and 50 epochs for single-frame and LSTM models respectively.

Qualitative Results: Figure 9 shows the predictions of the single-frame model, and the corresponding ground-truth. We use ground-truth hand and object poses for visualization purposes. They are not the inputs of our network. Here we focus on the colors on the meshes; Brighter colors represent closer distances in the interaction fields. The figure shows the feasibility of the task because the predictions correlate well with the ground-truth.

4. Metrics and Experiments

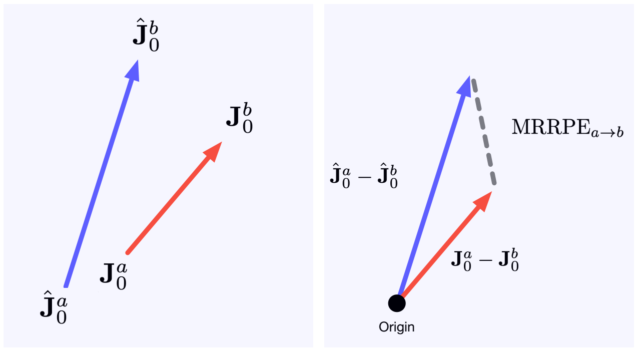
4.1. Metric Details

Acceleration Error (ACC): Following [12], we report acceleration error in m/s^2 to measure the smoothness of consistent motion reconstruction, and interaction field estimation. Formally, suppose $\hat{\mathbf{h}}_i^t \in \mathbb{R}^d$ is the predicted vertex (or distance value) i at frame t of a hand; $\mathbf{h}_i^t$ is the corresponding ground-truth. We compute the corresponding acceleration vector $\hat{\mathbf{u}}_i^t \in \mathbb{R}^d$ of $\hat{\mathbf{h}}_i^t$. Similarly, we compute the acceleration vector $\mathbf{u}_i^t$ for the ground-truth. The acceleration error for a hand is computed as:

$$\frac{1}{TV_h}\sum_{t=1}^{T}\sum_{i=1}^{V_h}\left\|\hat{\mathbf{u}}_i^t - \mathbf{u}_i^t\right\| \tag{5}$$

	Contact and Relative Position		Motion		Hand	Object	
Object	CDev$_{ho}$ $[mm]\downarrow$	MRRPE$_{rl/ro}$ $[mm]\downarrow$	MDev$_{ho}$ $[mm]\downarrow$	ACC$_{h/o}$ $[m/s^2]\downarrow$	MPJPE$_h$ $[mm]\downarrow$	AAE $[°]\downarrow$	Success Rate $[\%]\uparrow$
Notebook	37.4	47.7/39.8	9.9	5.0/6.5	20.8	3.3	80.4
Box	47.5	66.3/49.2	10.6	5.5/6.7	24.5	1.3	88.2
Espresso machine	48.9	52.5/46.2	9.5	4.8/5.0	24.5	11.0	81.0
Waffle iron	41.8	43.3/39.0	14.6	5.6/7.9	21.3	3.1	74.0
Laptop	42.6	54.7/40.5	12.8	5.2/7.2	21.7	1.7	84.4
Phone	29.5	42.2/31.1	7.5	4.6/7.2	18.8	3.9	62.3
Capsule Machine	30.5	37.6/30.9	7.8	4.7/4.4	19.2	6.9	69.3
Mixer	34.5	41.2/33.9	8.6	4.8/5.3	21.3	2.6	78.3
Ketchup bottle	33.0	45.6/35.0	10.8	5.4/7.4	20.7	7.0	59.2
Scissors	25.6	39.7/22.2	5.8	4.1/5.0	17.7	10.5	50.1
Microwave	60.8	62.6/41.9	9.3	5.2/5.2	26.0	7.3	74.3

Table 6. **Detailed breakdown on test set evaluation per object**. Here we provide the detailed breakdown of the test set evaluation according to each object. For each metric, we use red to denote the object with the highest error; we use blue to denote the lowest error.



(a) Predicted and ground-truth roots of entities *a* and *b*

(b) MRRPE for entities *a* and *b*

Figure 6. **An illustration of the MRRPE$_{a\rightarrow b}$ metric**. (a) The predicted roots of entities a and b are denoted by $\hat{\mathbf{J}}_0^a$ and $\hat{\mathbf{J}}_0^b$, and $\mathbf{J}_0^a$ and $\mathbf{J}_0^b$ are the corresponding ground-truth. (b) Subtract a by b; MRRPE$_{a\rightarrow b} \in \mathbb{R}$ is indicated by the dash line.

where V_h, T, d are the number of hand vertices, the sequence length, and the number of dimension for the prediction of a task. To compute the acceleration, we use centered difference:

$$u_i^t = \frac{h_i^{t-1} - 2h_i^t + h_i^{t+1}}{w^2} \tag{6}$$

where $w = 1/30s$ is the stencil width of 30-FPS videos. Note that previous methods [12, 28] computing the acceleration errors did not divide the error by w^2, leading to significantly smaller errors. The prediction dimension (d) for the reconstruction task, and the interaction field task are 3 and 1 respectively. For the former, we use root-relative vertices. We compute the acceleration of the object in the same way.

Average Articulation Error (AAE): Suppose $\omega^t \in \mathbb{R}$ and $\hat{\omega}^t \in \mathbb{R}$ are the predicted and ground-truth object articulation at frame t, the average articulation error is defined as

$$\frac{1}{T}\sum_{t=1}^{T} |\omega^t - \hat{\omega}^t| \tag{7}$$

where T is the number of frames.

	Contact and Relative Position		Motion		Hand	Object	
Size	CDev$_{ho}$	MRRPE$_{rl/ro}$	MDev$_{ho}$	ACC$_{h/o}$	MPJPE$_h$	AAE	Success R.
5	39.9	48.0/37.4	9.3	6.2/8.2	23.3	6.1	72.7
11	**39.0**	**47.0/36.6**	**8.8**	**6.1**/7.7	**22.8**	**5.8**	**74.6**
15	39.7	47.8/36.8	9.0	6.2/**7.6**	22.9	**5.8**	74.4

Table 7. **Effects of window size on ArcticNet-LSTM**. Here we ablate the effect of window size on our model on the validation set.

	Contact and Relative Position		Motion		Hand	Object	
CDev loss	CDev$_{ho}$	MRRPE$_{rl/ro}$	MDev$_{ho}$	ACC$_{h/o}$	MPJPE$_h$	AAE	Success R.
✗	49.0	53.1/45.6	11.9	7.3/10.1	**23.0**	6.1	71.3
✓	**41.9**	**50.1/37.6**	**10.4**	**7.3/9.8**	23.1	**5.9**	**71.8**

Table 8. **Effects of contact deviation (CDev) as a training loss**. Here we ablate the effect of the contact deviation metric as a loss on ArcticNet-SF on the validation set.

Mean Relative-Root Position Error (MRRPE): Following [6, 17], to measure the relative root translation between two entities a and b in the scene (a hand or an object),

$$\text{MRRPE}_{a\rightarrow b} = \left\| \left(\mathbf{J}_0^a - \mathbf{J}_0^b\right) - \left(\hat{\mathbf{J}}_0^a - \hat{\mathbf{J}}_0^b\right) \right\|_2, \tag{8}$$

where $a \in \{l, r, o\}$ and $b \in \{l, r, o\}$ and l, r, o denote the left hand, right hand, and the object, $\mathbf{J}_0 \in \mathbb{R}^3$ is the ground-truth root joint location and $\hat{\mathbf{J}}_0$ the predicted one. Figure 6 shows a visualization of the metric. Suppose we want to compute MRRPE$_{a\rightarrow b}$, and $\hat{\mathbf{J}}_0^a$ and $\hat{\mathbf{J}}_0^b$ denote the predicted roots for entities a and b and the notations without "hat" are the ground-truth. The MRRPE value is the distance indicated in the dash line.

4.2. Ablation and Analysis

Detailed analysis on test set: To see a performance breakdown per object, Table 6 shows the evaluation of the ArcticNet-LSTM model evaluated on the test set of the allocentric split. We use red to denote with the worst value for each metric and blue to denote the best value. We can see that the microwave is the hardest object in terms of hand reconstruction (see MPJPE) and contact (see CDev$_{ho}$). This

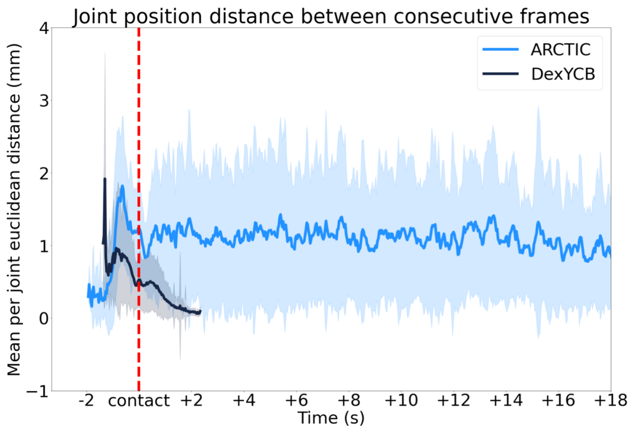


Figure 7. Changes in hand poses after first contact.

is because when opening the microwave door, the fingers are often heavily occluded. In contrast, the scissors has the lowest hand reconstruction error because it is a small object. In terms of estimating the articulation angle, the box, however, has the smallest error (see AAE). We hypothesize that the articulation angle is easy to observe because the box is the largest object. In contrast, the espresso machine has a small handle, which can be occluded heavily, so it has the highest AAE error. Finally, the scissors is the hardest object to reconstruct its pose (see success rate) because it is a small object and in some view its dark texture is similar to the background color.

Window size for ArcticNet-LSTM: Table 7 shows the effect of window size on ArcticNet-LSTM in the validation set. With more frames in a window, overall the model has better performance. To balance performance and efficiency, we use a window size of 11 for our model.

Contact deviation (CDev) as training loss: Table 8 shows the effect of the contact deviation metric as a training loss to encourage hand-object contact using the allocentric protocol. The results are evaluated on the validation set. In this ablation, we train the ArcticNet-SF by turning on and off the contact deviation loss. The loss is applied between left-hand/object, and right-hand/object as described in Sec. 3. Results show that the CDev loss improves hand-object contact indicated by the CDev$_{ho}$ metric.

Number of views for ArcticNet-SF: Since ARCTIC has only 1 egocentric view, we ablate the effect of the number of allocentric views. When trained with 2, 4, 6, 8 views, randomly selected, MPJPEs for hands are $49.0mm$, $35.0mm$, $25.8mm$, $23.4mm$; the object success rates are 40.8%, 57.0%, 62.1%, 68.6% on the validation set.

Quantifying dexterous motion: Existing datasets do not show significant changes in hand pose. In particular, poses in ContactPose are fixed relative to the object while DexYCB poses do not change much once contact is established (see Fig. 7). The figure plots the relative change in 3D joints across consecutive frames, without global translation and rotation. The vertical dashed line indicates the first contact. Fig. 2 (main paper) shows that ARCTIC has more diversity in hand poses and contact patterns, resulting from dexterous manipulation, compared to other datasets.

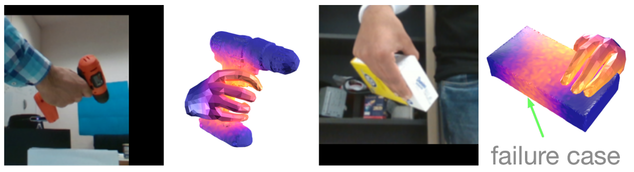


Figure 8. **Interaction field estimation on HO3D**

Interaction field estimation on HO3D: Our proposed task can be applied to existing hand datasets with rigid objects. To show this, we trained InterField-SF on HO3D. Fig. 8 shows qualitative results. However, we note that existing hand-object datasets have similar hand poses within each sequence (thus, similar contact) and fewer training images, which are easy to overfit to (see Fig. 8 failure case); ARCTIC is large-scale and it is more challenging due to more dynamic interaction (changing poses and contact) and thus will help in fostering future research.

Improve hand and rigid objects with ARCTIC: We pre-train ArcticNet on ARCTIC, finetune on HO3D (hand + rigid objects). This model is compared to a model trained only on HO3D. Following the HO3D protocol, pre-training on ARCTIC improves MPJPE (scale-translation aligned) errors by 9.2%. For the object, the vertex-to-vertex error (root aligned) improves by 7.1%. This shows that articulated hand-object data benefits hand and rigid object reconstruction.

5. Visualizing ARCTIC

To visualize the our 3D annotation in the dataset, Fig. 10 shows random samples of the 3D meshes of hands and objects overlaid on the images in our ARCTIC dataset. See the video on our project page for our rendered sequences.

6. Discussions and Limitations

We introduce ARCTIC, the first dataset for two hands dexterously manipulating articulated objects and baselines for the task of consistent motion reconstruction, and interaction field estimation. Being a first step, our work is not without limitations.

Known object models in ArcticNet: Similar to existing methods [3, 8, 9, 24], one limitation of our baselines is that they assume known object models. We view articulated 3D shape estimation of unknown objects as an orthogonal problem on which the field is making progress. Now that we have showed the feasibility of inferring hand-object interaction for such objects, future work should bring together our method with 3D articulated object inference. This is challenging and we believe it is critical to make progress on sub-problems, for which ARCTIC can be leveraged.

Toy objects: Some of our objects are toys, which are not to scale and lack some of the visual complexity of real objects. However, the aim of ARCTIC is to study the physical dynamics between hand-object motions.

SMPL-X for capturing contact: Since we use SMPL-X/MANO as our human representation, the human geometry does not capture skin deformation during contact. While a deformable human body/hand model would be ideal for capturing true contact, developing such models is not a goal of ARCTIC. Further, marker and image data ARCTIC can be used to fit a deformable model if developed. Our 3D annotation and contact capture pipeline follows GRAB [21]. In particular, we use MoSh++ to fit SMPL-X to the markers, producing highly accurate fits. We adapt the contact capture pipeline from GRAB. GRAB contact labels are widely used in the community to support projects such as [7, 20, 26].

Markers on hands in our RGB images: We use optical marker-based capture to provide accurate hand and object poses, thus potentially introducing label noise. However, our hand markers are minimally intrusive (1.5mm in radius) and barely visible when images are resized for inputs.

Degree of freedom in our objects: We construct ARCTIC with objects of 1 DoF. This is because many items designed for human interaction often have a single axis of rotation as they are easy to produce and intuitive to manipulate (*e.g*., doors, refrigerators, ovens, pliers, *etc*.). Thus, objects in ARCTIC are representative of a broad class of objects found in homes and businesses. Importantly, reconstructing interaction with such objects involves occlusion, depth ambiguity, and contact estimation. These issues also apply to objects with more DoF. Further, we capture more diverse hand poses and more challenging hand-object interactions compared to existing hand-object datasets. Future work should expand the number and complexity of objects to further study the problems of depth ambiguity and occlusion.

Approval for human subject data: Subject data was collected with written, prior, informed consent and the data collection was reviewed by the university ethics board.

References

[1] Adnane Boukhayma, Rodrigo de Bem, and Philip H. S. Torr. 3D hand shape and pose from images in the wild. In *Computer Vision and Pattern Recognition (CVPR)*, pages 10843–10852, 2019.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. Ieee, 2009.

[3] Bardia Doosti, Shujon Naha, Majid Mirbagheri, and David J. Crandall. HOPE-Net: A graph-based model for hand-object pose estimation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 6607–6616, 2020.

[4] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y. Zeevi. The farthest point strategy for progressive image sampling. In *International Conference on Pattern Recognition (ICPR)*, pages 93–97, 1994.

[5] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y. Zeevi. The farthest point strategy for progressive image sampling. *Transactions on Image Processing (TIP)*, 6(9):1305–1315, 1997.

[6] Zicong Fan, Adrian Spurr, Muhammed Kocabas, Siyu Tang, Michael J. Black, and Otmar Hilliges. Learning to disambiguate strongly interacting hands via probabilistic per-pixel part segmentation. In *International Conference on 3D Vision (3DV)*, pages 1–10, 2021.

[7] Patrick Grady, Chengcheng Tang, Christopher D. Twigg, Minh Vo, Samarth Brahmbhatt, and Charles C. Kemp. ContactOpt: Optimizing contact to improve grasps. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1471–1481, 2021.

[8] Yana Hasson, Bugra Tekin, Federica Bogo, Ivan Laptev, Marc Pollefeys, and Cordelia Schmid. Leveraging photometric consistency over time for sparsely supervised hand-object reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*, pages 568–577, 2020.

[9] Yana Hasson, Gül Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *Computer Vision and Pattern Recognition (CVPR)*, pages 11807–11816, 2019.

[10] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Computer Vision and Pattern Recognition (CVPR)*, pages 7122–7131, 2018.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[12] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. VIBE: Video inference for human body pose and shape estimation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 5253–5263, 2020.

[13] Muhammed Kocabas, Chun-Hao P. Huang, Otmar Hilliges, and Michael J. Black. PARE: Part attention regressor for 3D human body estimation. In *International Conference on Computer Vision (ICCV)*, pages 11127–11137, 2021.

[14] Muhammed Kocabas, Chun-Hao P. Huang, Joachim Tesch, Lea Müller, Otmar Hilliges, and Michael J. Black. SPEC: Seeing people in the wild with an estimated camera. In *International Conference on Computer Vision (ICCV)*, pages 11035–11045, 2021.

[15] 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 *Computer Vision and Pattern Recognition (CVPR)*, pages 21013–21022, 2022.

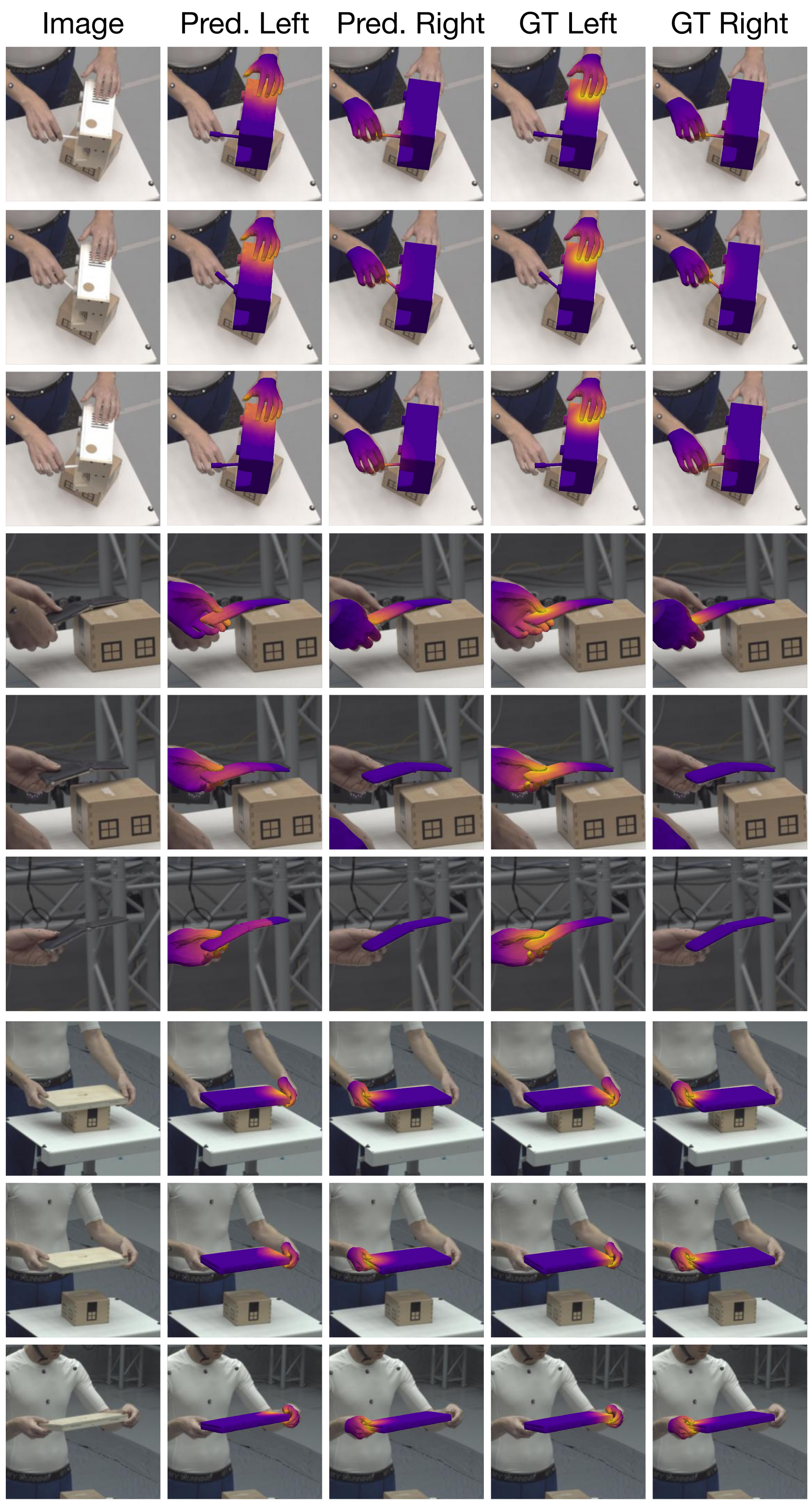


Figure 9. **Qualitative results of InterField-SF**. Best viewed in color and zoomed in.

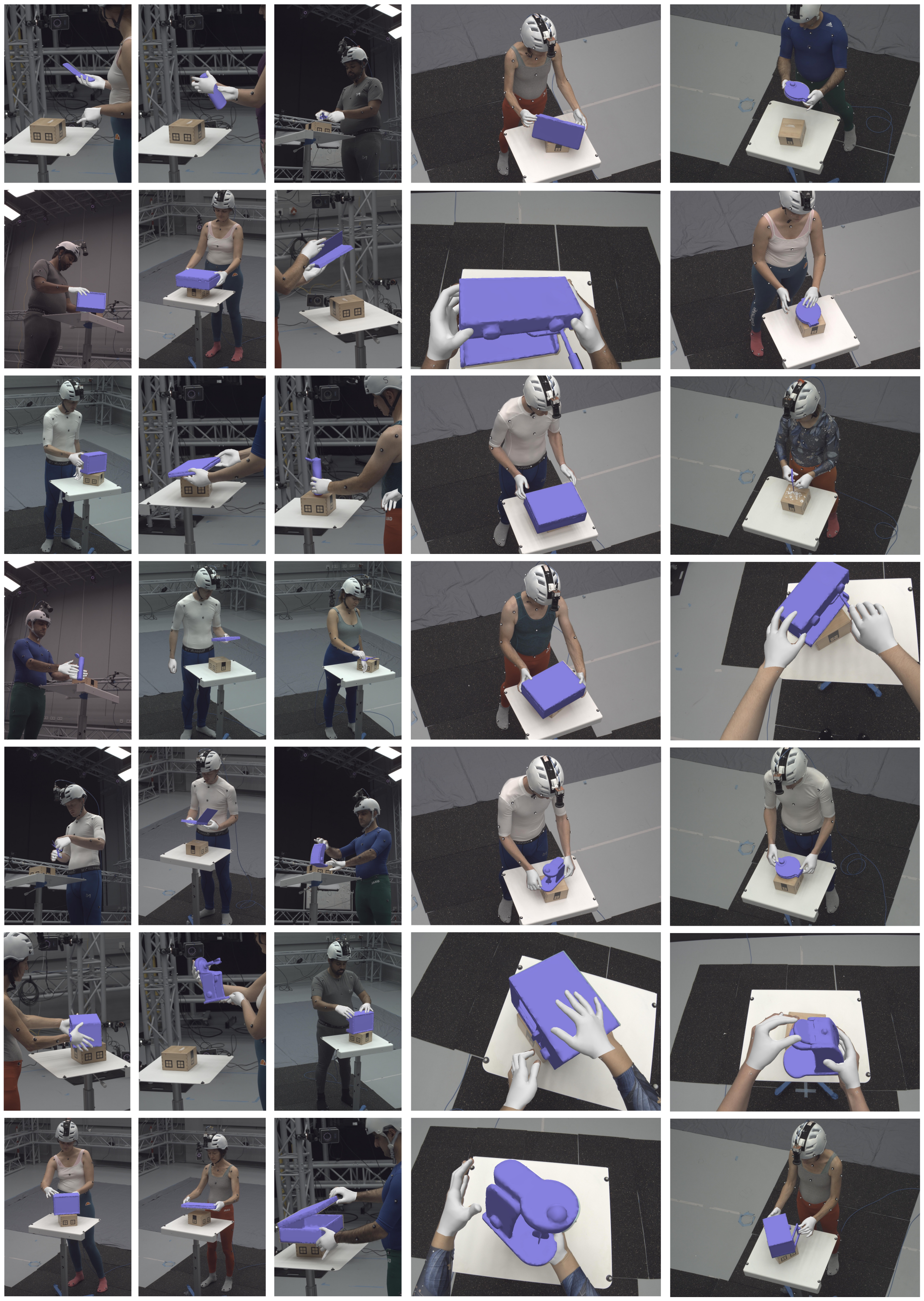


Figure 10. **Overlay of ARCTIC ground-truth** We overlay the ground-truth in our dataset. Examples here are randomly sampled from ARCTIC. Note that although the RGB images contain both human bodies and hands, the hand region is clearly visible when zoomed in, thanks to our high resolution RGB images.

[16] Frank Michel, Alexander Krull, Eric Brachmann, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Pose estimation of kinematic chain instances via object coordinate regression. In *British Machine Vision Conference (BMVC)*, pages 181.1–181.11, 2015.

[17] Gyeongsik Moon, Shoou-I Yu, He Wen, Takaaki Shiratori, and Kyoung Mu Lee. InterHand2.6M: A dataset and baseline for 3D interacting hand pose estimation from a single RGB image. In *European Conference on Computer Vision (ECCV)*, volume 12365, pages 548–564, 2020.

[18] István Sárándi, Timm Linder, Kai O. Arras, and Bastian Leibe. Metric-scale truncation-robust heatmaps for 3D human pose estimation. In *International Conference on Automatic Face & Gesture Recognition (FG)*, pages 407–414, 2020.

[19] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[20] Omid Taheri, Vassileios Choutas, Michael J. Black, and Dimitrios Tzionas. GOAL: Generating 4D whole-body motion for hand-object grasping. In *Computer Vision and Pattern Recognition (CVPR)*, pages 13253–13263, 2022.

[21] Omid Taheri, Nima Ghorbani, Michael J. Black, and Dimitrios Tzionas. GRAB: A dataset of whole-body human grasping of objects. In *European Conference on Computer Vision (ECCV)*, volume 12349, pages 581–600, 2020.

[22] 3dMDhand system series. `https://3dmd.com/products/`.

[23] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *Computer Vision and Pattern Recognition (CVPR)*, pages 11094–11104, 2020.

[24] Lixin Yang, Xinyu Zhan, Kailin Li, Wenqiang Xu, Jiefeng Li, and Cewu Lu. CPF: Learning a contact potential field to model the hand-object interaction. In *International Conference on Computer Vision (ICCV)*, pages 11097–11106, 2021.

[25] Xiong Zhang, Qiang Li, Hong Mo, Wenbo Zhang, and Wen Zheng. End-to-end hand mesh recovery from a monocular RGB image. In *International Conference on Computer Vision (ICCV)*, pages 2354–2364, 2019.

[26] Keyang Zhou, Bharat Lal Bhatnagar, Jan Eric Lenssen, and Gerard Pons-Moll. TOCH: Spatio-temporal object-to-hand correspondence for motion refinement. In *European Conference on Computer Vision (ECCV)*, volume 13663, pages 1–19, 2022.

[27] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 5745–5753, 2019.

[28] Andrea Ziani, Zicong Fan, Muhammed Kocabas, Sammy Christen, and Otmar Hilliges. TempCLR: Reconstructing hands via time-coherent contrastive learning. In *International Conference on 3D Vision (3DV)*, pages 627–636, 2022.