MobileOcc: A Human-Aware Semantic Occupancy Dataset for Mobile Robots

Junseo Kim[*1], Guido Dumont[*1], Xinyu Gao[*1], Gang Chen[*1], Holger Caesar[1], and Javier Alonso-Mora[1]

Supplementary Material

1 Visibility filtering

We follow the notation introduced in **??** of the main paper: SMPL parameters $(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{t}_{\text{cam}})$, mesh $\mathcal{M}$ with visible subset $\mathcal{V} \subset \mathcal{M}$, LiDAR point cloud $\mathcal{P}$, and camera intrinsics $\mathbf{K}$. Our visibility filtering selects a subset of mesh vertices that i) are geometrically visible from the camera and ii) belong to body parts that are likely unoccluded in the image. The resulting visible subset $\mathcal{V}$ is then used for ICP registration and for the 3D LiDAR alignment loss $\mathcal{L}_{3D}$.

1.1 Backface culling

The SMPL mesh $\mathcal{M}$ consists of triangular faces, each with three vertices in 3D. For each face f, we compute i) a unit face normal $\mathbf{n}_f$ that indicates which direction the face is pointing, and ii) a representative point $\mathbf{c}_f$ at the center (centroid) of the triangle. Let $\mathbf{o}$ denote the 3D position of the camera. The viewing direction from the camera to face f is then

$$\mathbf{p}_f = \frac{\mathbf{c}_f - \mathbf{o}}{\|\mathbf{c}_f - \mathbf{o}\|_2}. \tag{1}$$

We classify a face as front-facing if the angle between its normal and the viewing direction is sufficiently small. We implement this using the inner product between the normal and the viewing direction:

$$\mathbf{n}_f^\top \mathbf{p}_f < w, \tag{2}$$

where w is a backface-culling threshold (tuned per dataset, see Tab. 1). All faces that satisfy this condition are considered front-facing, and all vertices that belong to at least one such face form a candidate visible set $\mathcal{V}_{\text{front}} \subset \mathcal{M}$.

This step removes mesh regions oriented away from the camera. It therefore should not be matched to LiDAR points, thereby reducing the influence of faces that are clearly not visible from the current viewpoint.

1.2 Body-part filtering from 2D keypoints

Backface culling alone does not handle self-occlusion (e.g., the arm on the far side of the body) or occlusion by other objects. To further restrict the mesh to the observed regions, we use the same 2D keypoints and confidence scores as in the 2D joint reprojection loss, $\mathcal{L}_J$.

Let $\mathbf{J}_i^{\text{est}}$ denote the detected 2D joints and w_i their confidence scores. For each SMPL body part b (e.g., torso, left upper arm, right lower leg), we define a small set of 2D joints $\mathcal{J}_b$ that are informative for that part (for example, shoulder and elbow for an upper arm). A body part is marked as occluded if all of its associated joints have low confidence:

$$\max_{i \in \mathcal{J}_b} w_i < J_{\text{conf}}, \tag{3}$$

where J_{conf} is a dataset-specific threshold, listed in Tab. 1.

For each occluded body part, we remove all mesh faces whose vertices belong to that part from the candidate visible set $\mathcal{V}_{\text{front}}$ obtained by backface culling. Since the SMPL template provides a fixed correspondence between mesh faces and semantic body parts, this mapping is known in advance and does not depend on pose or shape. The remaining faces define a refined visible mesh that is consistent with both the camera viewpoint and the 2D keypoint evidence. This body-part filtering improves robustness to occluded limbs by preventing LiDAR points from being matched to mesh regions that are not supported by the image.

1.3 Final visible vertex set

Combining backface culling and body-part filtering yields our final set of visible mesh vertices $\mathcal{V} \subset \mathcal{M}$. By construction,

- $\mathcal{V}$ contains only vertices that belong to front-facing faces, and
- vertices assigned to body parts marked as occluded by the 2D keypoints are removed.

We treat $\mathcal{V}$ as the subset of the mesh that is both visible in the image and observable by the LiDAR sensor. All subsequent 3D processing is restricted to $\mathcal{V}$. Focusing on $\mathcal{V}$ instead of the full mesh reduces the influence of self-occluded or truncated limbs and leads to more stable LiDAR-mesh alignment in scenes with occlusions.

2 Scalar Weights and Optimization

The mesh refinement in **??** of the main paper minimizes the multi-term objective in **??**, combining 2D joint reprojection $\mathcal{L}_J$, 3D LiDAR alignment $\mathcal{L}_{3D}$, pose and shape priors $\mathcal{L}_\theta$ and $\mathcal{L}_\beta$, an anti-hyperextension prior $\mathcal{L}_a$, and an occlusion-aware pose consistency term $\mathcal{L}_{\theta,\text{occ}}$. All terms are fully differentiable with respect to $(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{t}_{\text{cam}})$.

Our optimization scheme contains several scalar hyperparameters:

- Loss weights λ_θ, λ_a, λ_β, λ_{3D}, and λ_{occ}.
- The Geman–McClure scale ρ used in L_J.
- The backface culling threshold w used in the visibility filter.
- The joint confidence threshold J_{conf} used for part-level occlusion reasoning.

These hyperparameters influence the trade-off between image alignment, LiDAR alignment, and prior regularization. To obtain the best configurations, we perform Bayesian Optimization on a subset that is held out from the final evaluation for each dataset. For each trial, we run the full optimization pipeline with a candidate parameter configuration and evaluate the resulting meshes using the relevant 3D metrics (e.g., PVE, MPJPE, PA-MPJPE). Bayesian Optimization maintains a surrogate model of the objective over the hyperparameter space and iteratively proposes new configurations that balance exploration and exploitation. In practice, we run 100 iterations per dataset and select the best-performing configuration on the subset.

Tab. 1 lists the resulting values for all datasets used in our experiments. Note that the UT Campus dataset does not provide ground-truth SMPL annotations; for this dataset, we select hyperparameters based on qualitative inspection of the resulting meshes.

Table 1: Sub-optimal scalar weights and thresholds per dataset.

Dataset	ρ	λ_θ	λ_a	λ_β	λ_{3D}	λ_{occ}	w	J_{conf}
3DPW	100	2.2	11.0	5.0	800	35	0.2	0.6
SLOPER4D	100	0.75	8.5	1.0	500	55	0.2	0.7
HumanM3	100	1.0	3.0	17.5	600	135	−1.0	0.6
UT Campus	100	1.4	10.0	10.0	300	50	0.2	0.7

In all experiments, we start from the initial mesh prediction provided by the base HMR model and run gradient descent on $(\beta, \theta, t_{\text{cam}})$ until convergence or a fixed iteration budget is reached. The chosen hyperparameters ensure that the optimization improves both image and LiDAR alignment while keeping poses and shapes within a plausible region of the SMPL space and preventing excessive drift of occluded limbs.

3 Synthetic LiDAR sweeps

For datasets without real LiDAR (3DPW [**?**]), we simulate LiDAR sweeps directly on the ground-truth SMPL meshes to study how LiDAR characteristics affect our method. For each camera frame, we place a virtual LiDAR at the camera center and emit rays based on the vertical and horizontal angular resolutions of an Ouster-style spinning sensor. Each ray intersects the SMPL mesh to obtain a 3D return, analogous to a time-of-flight measurement. If a ray does not hit the mesh within the valid range, it produces no return.

We model three virtual sensors with increasing angular resolutions: "Ouster-32", "Ouster-64", and "Ouster-128". For each simulated return, we add range and angular perturbations and randomly drop individual points to mimic missed detections. All rays are restricted to the camera frustum, so the simulated LiDAR only observes the same region as the RGB camera. Since 3DPW does not contain full 3D scene geometry, occlusions from other objects are not simulated. Instead, our visibility filter (Sec. 1) discards mesh regions that are likely occluded in the image.

The sensor configurations are summarized in Tab. 2. All three sensors share the same noise characteristics and valid distance range $[0.5, 90]$ m, but differ in vertical and horizontal resolution. We use these simulated sweeps as input to our pipeline, keeping all other components identical to the real-LiDAR setting. This design allows us to isolate the impact of LiDAR density and noise on performance.

Table 2: Specifications of the simulated LiDAR sensors used on 3DPW [**?**]. Resolution is given in vertical and horizontal channels, range noise mean/std are in millimeters, angular noise mean/std are in degrees, and range min/max are in meters. All sensors use the same noise levels and a fixed dropout probability of 10%.

Nr.	*Type*	Resolution		Range noise		Angular noise		Range	
		Vert.	Hor.	mean	std	mean	std	min	max
1.	Ouster-32	32	512	±25.0	10.0	0.0	0.01	0.5	90.0
2.	Ouster-64	64	1024	±25.0	10.0	0.0	0.01	0.5	90.0
3.	Ouster-128	128	2048	±25.0	10.0	0.0	0.01	0.5	90.0

The simulated LiDAR sweeps approximate realistic Ouster sensors in terms of angular resolution, range noise, and dropout behavior, while being perfectly time-synchronized with the RGB images and SMPL ground truth. This provides a controlled setting to quantify how much additional 3D information is needed for our optimization to improve over purely image-based HMR.

3.1 Ablation study

We perform an ablation study on 3DPW [**?**] to quantify the contribution of each newly introduced component: i) the visibility filter, ii) the 3D LiDAR alignment term $\mathcal{L}_{3D}$, and iii) the occlusion-aware pose prior $\mathcal{L}_{\theta,\text{occ}}$. For each of the three simulated sensors, we run our method with one component removed at a time and compare PVE, MPJPE, and PA-MPJPE to the full model.

Tab. 3 focuses on the visibility filter. Since this component only affects frames where at least one body part is (partially) occluded, we construct a subset of 1,063 frames from 3DPW [**?**] that each contain at least one occluded limb (about 2.9% of the dataset). On this subset, we compare our full model with visibility

Table 3: Ablation study on the visibility filter for partially occluded body parts. "Ours" denotes the full model with the visibility filter, while "No visibility filtering" disables this component.

		Ours		No visibility filtering			
Sensor	Metric	mean	std	mean	std	t-value	p-value
Ouster-32	PVE ↓	82.0	50.9	84.3	60.2	9.512	0.3416
	MPJPE ↓	62.9	42.6	64.9	52.7	9.623	0.3360
	PA-MPJPE ↓	50.2	35.0	51.5	42.3	7.720	0.4402
Ouster-64	PVE ↓	66.9	40.0	69.3	47.6	12.585	0.2082
	MPJPE ↓	48.2	31.0	50.4	36.4	15.002	0.1337
	PA-MPJPE ↓	41.1	29.4	42.3	34.8	8.588	0.3905
Ouster-128	PVE ↓	58.7	35.4	60.2	42.2	8.879	0.3747
	MPJPE ↓	43.6	26.4	45.3	35.6	12.506	0.2112
	PA-MPJPE ↓	37.2	27.1	38.7	33.5	11.350	0.2565

filtering against a variant in which the filter is disabled. Across all three simulated sensors, removing the visibility filter consistently increases PVE, MPJPE, and PA-MPJPE, although the numerical differences are modest. This confirms that masking out occluded regions helps in precisely those challenging cases, while having little effect on fully visible poses.

Tab. 4 investigates the two new loss terms from **??**: the occlusion-aware pose prior $\mathcal{L}_{\theta,\text{occ}}$ and the 3D LiDAR alignment term $\mathcal{L}_{3D}$. Here, we evaluate on the full 3DPW test set (35,515 frames). For each sensor, we report the performance of the full model ("Ours") and compare it to i) a variant without $\mathcal{L}_{\theta,\text{occ}}$ and ii) a variant without $\mathcal{L}_{3D}$.

The 3D term $\mathcal{L}_{3D}$ has by far the most significant impact. Removing it roughly doubles the PVE and MPJPE across all three sensors, bringing performance close to that of using only ICP alignment without any LiDAR-aware refinement. This confirms that the LiDAR–mesh Chamfer term is crucial for correcting local pose and shape errors beyond what ICP alone can provide.

The occlusion-aware pose prior $\mathcal{L}_{\theta,\text{occ}}$ yields a more minor but consistent gain: disabling this term slightly worsens all metrics, indicating that softly constraining occluded joints toward their initial configuration helps prevent unrealistic drift while still allowing visible joints to move freely.

4 Baseline Details

BEVDet4D. The public configuration uses an effective batch size of 64 (8×8), a learning rate of 2×10^{-4}, and 20 epochs. We retain these settings but train on a single A40 using batch 8 with 8-step gradient accumulation, and shorten training to 15 epochs. The BEV and depth ranges are $x \in [0, 12.8]\,\text{m}$, $y \in [-6.4, 6.4]\,\text{m}$, and depth $[1, 15]\,\text{m}$ with $0.5\,\text{m}$ bins.

Table 4: Ablation study on the proposed loss terms in the objective function (**??**), evaluated on the full 3DPW [**?**] test set (35,515 frames). Removing the occlusion-aware pose prior $\mathcal{L}_{\theta,\text{occ}}$ slightly degrades performance, while removing the 3D LiDAR alignment term $\mathcal{L}_{3D}$ leads to a drastic increase in error across all sensors.

		Ours		No $\mathcal{L}_{\theta,\text{occ}}$		No $\mathcal{L}_{3D}$	
Sensor	Metric	mean	std	mean	std	mean	std
Ouster-32	PVE ↓	**73.2**	50.9	75.9	75.9	171.2	113.1
	MPJPE ↓	**57.0**	42.6	58.2	43.2	159.3	113.2
	PA-MPJPE ↓	**47.6**	35.0	48.9	35.5	49.6	22.1
Ouster-64	PVE ↓	**57.2**	40.0	60.1	41.3	173.2	117.4
	MPJPE ↓	**43.9**	31.0	45.2	31.8	161.7	117.8
	PA-MPJPE ↓	**38.5**	29.4	40.4	31.0	49.2	21.6
Ouster-128	PVE ↓	**50.5**	35.4	52.4	36.8	174.9	121.4
	MPJPE ↓	**39.1**	26.4	40.3	27.6	163.4	121.8
	PA-MPJPE ↓	**35.1**	27.1	36.6	28.9	49.4	23.0

FlashOcc and its variants. All variants are initialized from our BEVDet4D (1f) checkpoint (12 epochs). The original setup uses LR 1×10^{-4}, effective batch 16 (4×4), and 24 epochs; we match the batch size on a single A40 via batch 4 with 4-step accumulation and train for 15 epochs. The occupancy volume follows our final evaluation grid: $x \in [0.4, 10.0]\,\text{m}$, $y \in [-4.8, 4.8]\,\text{m}$, $z \in [-1.0, 3.8]\,\text{m}$ at 0.2 m resolution, with depth bins $[1, 12]\,\text{m}$ at 0.5 m.

VoxFormer. For VoxFormer-T, we keep the public training settings (20 epochs and learning rate 2×10^{-4}). The original configuration uses a batch size of 1 on 8 GPUs (effective batch size 8); on our single A40, we use a batch size of 1 with 8-step gradient accumulation to match.

Input processing and augmentation. All methods use the same image pipeline: input images of 1024×1224 (H×W) are vertically cropped from rows 144 to 1008 (yielding 864×1224) and then resized to 384×544. BEVDet4D and the FlashOcc family use the default image augmentations from their public code (no BEV-space augmentation). For fairness, we also implement a random horizontal flip for VoxFormer in the image space.

EMA and checkpoints. The public VoxFormer code does not use an exponential moving average (EMA), so we report results from the final checkpoint. BEVDet4D and the FlashOcc family use EMA in their official implementations, and we therefore evaluate these baselines using their EMA weights.