Robust 6DoF Pose Estimation Against Depth Noise and a Comprehensive Evaluation on a Mobile Dataset

Zixun Huang[1*] **Keling Yao**[2*] **Seth Z. Zhao**[3] **Chuanyu Pan**[1] **Allen Y. Yang**[1]
[1]UC Berkeley [2]CMU [3]UCLA

Abstract

Robust 6DoF pose estimation with mobile devices is the foundation for applications in robotics, augmented reality, and digital twin localization. In this paper, we extensively investigate the robustness of existing RGBD-based 6DoF pose estimation methods against varying levels of depth sensor noise. We highlight that existing 6DoF pose estimation methods suffer significant performance discrepancies due to depth measurement inaccuracies. In response to the robustness issue, we present a simple and effective transformer-based 6DoF pose estimation approach called DTTDNet[1]*, featuring a novel geometric feature filtering module and a Chamfer distance loss for training. Moreover, we advance the field of robust 6DoF pose estimation and introduce a new dataset – Digital Twin Tracking Dataset Mobile (DTTD-Mobile), tailored for digital twin object tracking with noisy depth data from the mobile RGBD sensor suite of the Apple iPhone 14 Pro. Extensive experiments demonstrate that DTTDNet significantly outperforms state-of-the-art methods at least 4.32, up to 60.74 points in ADD metrics on the DTTD-Mobile. More importantly, our approach exhibits superior robustness to varying levels of measurement noise, setting a new benchmark for robustness to measurement noise. The project page is publicly available at https://openark-berkeley.github.io/DTTDNet/.*

1. Introduction

Six-degrees-of-freedom (6DoF) object pose estimation aims at determining the position and orientation of an object in 3D space. In contrast to the more matured technology of camera tracking in static settings known as visual odometry or simultaneous localization and mapping [3, 17, 29, 32], identifying the relative position and orientation of one or more objects with respect to the user's ego position is a core function essential for ensuring a high-quality user experience in applications like augmented reality (AR). In the most general setting, each object with respect to the ego position may undergo independent rigid-body motion, and the combined effect of overlaying multiple objects in the scene may also cause parts of the objects to be occluded from the measurement of the ego position. In this paper, the main topic of our investigation is to study the 6DoF pose estimation problem under a wide range of motion, occlusion, color, and lighting conditions, especially improving the accuracy and robustness of algorithms under novel data sensor properties. The dataset and proposed model are made publicly available at https://github.com/augcog/DTTD2.

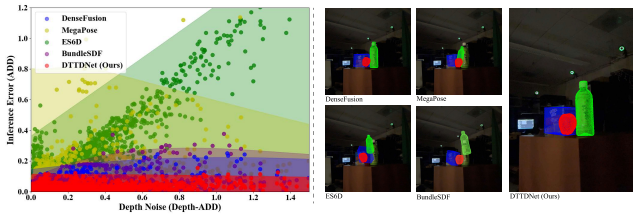

Figure 1. *Left*: Shadow plot of the relation between the **depth noise** (*depth-ADD*) and the **inference error** (ADD) of considered state-of-the-art methods and proposed DTTDNet. *Right*: Visualization of pose estimation results of baseline methods and proposed DTTDNet.

Recent advancements in the field of 6DoF pose estimation have primarily been motivated by deep neural network (DNN) approaches that advocate end-to-end training to carry out crucial tasks such as image semantic segmentation, object classification, and object pose estimation. Notable studies [10, 11, 16, 27, 35] have demonstrated the effectiveness of these pose estimation algorithms using established real-world 6DoF pose estimation datasets [13, 14, 23, 24, 26, 39]. However, it should be noted that these datasets primarily focus on robotic grasping tasks, and applying these solutions to environments served with mobile devices introduces a fresh set of challenges. A previous work [6] first studied this gap in the context of 6DoF pose

*Equal contributions. Contact: zixun@berkeley.edu.

[1]This work was previously presented as a non-archival poster at the 2024 ICML DMLR Workshop. This submission does not overlap with any archival publications.

estimation and replicated real-world digital-twin scenarios with varying levels of capture distances, lighting conditions, and object occlusions. It is important to mention that, this dataset was collected using Microsoft Azure Kinect, which may not be the most suitable camera platform for studying 3D localization under realistic mobile environments.

Alternatively, Apple has emerged as a strong proponent of utilizing RGB-D spatial sensors for mobile AR applications with the design of their iPhone Pro camera suite, such as on the latest Apple iPhone 14 Pro model. This particular smartphone is equipped with a rear-facing LiDAR depth sensor [2, 7, 15, 21, 38, 40], a critical component to achieving accurate and detailed 3D perception and spatial understanding. However, one distinguishing drawback of the iPhone LiDAR depth is the low resolution of the depth map produced by the iPhone ARKit [29], a 256×192 resolution compared to a 1280×720 depth map provided by the Microsoft Azure Kinect. This low resolution is exacerbated by large errors in the retrieved depth map, which is also observed in [1, 19]. The large amounts of errors (as shown in Fig. 5) in the iPhone data also pose challenges for researchers to develop a pose estimator that can correctly predict object poses that rely heavily on the observed depth map, which has not been particularly addressed in previous works [7, 27, 33, 35, 38, 40]. We will demonstrate this in following experiments (Table 3).

To investigate the 6DoF pose estimation problem under the most popular mobile depth sensor, namely, the Apple iPhone 14 Pro LiDAR, we propose an RGBD-based transformer model for 6DoF object pose estimation, which is designed to effectively handle inaccurate depth measurements and noise. As shown in Fig. 1, our method shows robustness against noisy depth input, while other baselines failed in such conditions. Meanwhile, we introduce DTTD-Mobile, a novel RGB-D dataset captured by iPhone 14 Pro, to bridge the gap of digital-twin pose estimation with mobile devices, allowing research into extending algorithms to iPhone data and analyzing the unique nature of iPhone depth sensors. Our contributions are summarized into three parts:

1. We propose a new transformer-based 6DoF pose estimator with depth-robust designs on modality fusion and training strategies, called DTTDNet. The new solution outperforms other state-of-the-art methods by a large margin especially in noisy depth conditions.
2. We introduce DTTD-Mobile as a novel digital-twin pose estimation dataset captured with mobile devices. We provide in-depth LiDAR depth analysis and evaluation metrics to illustrate the unique properties and complexities of mobile LiDAR data.
3. We conduct extensive experiments and ablation studies to demonstrate the efficacy of DTTDNet and shed light on how the depth robustifying module works.

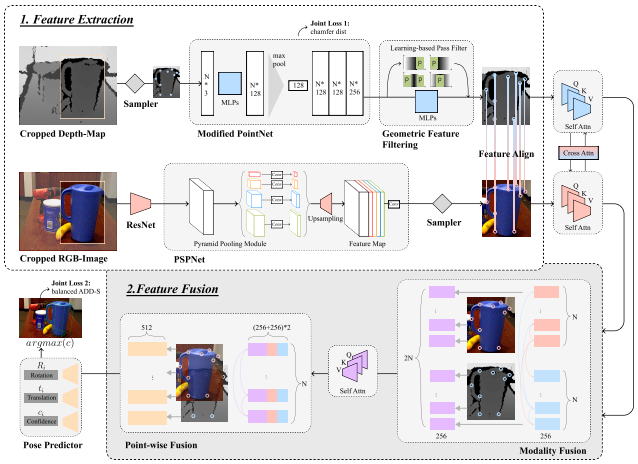

Figure 2. **Model Architecture Overview.** DTTDNet pipeline starts with segmented depth maps and cropped RGB images. The point cloud from the depth map and RGB colors are encoded and integrated point-wise. Extracted features are then fed into an attention-based two-stage fusion. Finally, the pose predictor produces point-wise predictions with both rotation and translation.

2. Methods

In this section, we elaborate on the details of our method. The objective is to estimate the 3D location and pose of a known object in the camera coordinates from the RGBD images. This position can be represented using homogeneous transformation matrix $p \in SE(3)$, which consists of a rotation matrix $R \in SO(3)$ and a translation matrix $t \in \mathbb{R}^3$, $p = [R|t]$. Section 2.1 describes our transformer-based model architecture. Section 2.2 introduces two depth robustifying modules on depth feature extractions, dedicated to geometric feature reconstruction and filtering. Section 2.3 illustrates our modality fusion design for the model to disregard significant noisy depth features. Finally, Section 2.4 describes our final learning objective.

2.1. Architecture Overview

Fig. 2 illustrates the overall architecture of the proposed DTTDNet. The DTTDNet pipeline takes segmented depth maps and cropped RGB images as input. It then obtains feature embedding for both RGB and depth images through separate CNN and point-cloud encoders on cropped RGB images and reconstructed point cloud corresponding to the cropped depth images.[2] Inspired by PSPNet [41], the image embedding network comprises a ResNet-18 encoder, which is then followed by 4 up-sampling layers acting as the decoder. It translates an image of size $H \times W \times 3$ into a $H \times W \times d_{rgb}$ embedding space. For depth feature extrac-

[2]We preprocessed the RGB and depth images to guarantee the pixel-level correspondence between the RGB image and the depth image. The preprocessing process is detailed in Section 3.

tion, we take segmented depth pixels and transform them into 3D point clouds with the camera intrinsic.

The 3D point clouds are initially processed using an auto-encoder inspired by the PointNet [30]. The PointNet-style encoding step aims to capture geometric representations in latent space in $\mathbb{R}^{d_1}$. In this context, the encoder component produces two sets of features: early-stage point-wise features in $\mathbb{R}^{N\times d_2}$ and global geometric features in $\mathbb{R}^{d_3}$. Subsequently, we add a decoder that is guided by a reference point set P to generate the predicted point cloud $\hat{P}$. Features extracted from the encoder are subsequently combined with the learned representations to create a new feature sequence with a dimension of $\mathbb{R}^{N\times d_{geo}}$, where $d_{geo} = d_1 + d_2 + d_3$.

This results in a sequence of geometric tokens with a length equal to the number of points N. Extracted RGB and depth features are then fed into a two-stage attention-based fusion block, which consists of modality fusion and point-wise fusion. Finally, the pose predictor produces point-wise predictions with both rotation and translation. The predictions are then voted based on unsupervised confidence scoring to get the final 6DoF pose estimate.

2.2. Design for Robustifying Depth Data

In this section, we will introduce two modules (Fig. 3) that enable the point-cloud encoder in DTTDNet to handle noisy and low-resolution LiDAR data robustly.

Chamfer Distance Loss. Past methods either treated the depth information directly as image channels [27] or directly extracted features from a point cloud for information extraction [35]. These methods underestimated the corruption of the depth data caused by noise and error during the data collection process. To address this, we first introduce a downstream task for point-cloud reconstruction and utilize the Chamfer distance as a loss function to assist our feature embedding in filtering out noise. The Chamfer distance loss is widely used for denoising in 3D point clouds [5, 12], and it is defined as the following equation between two point clouds $P \in \mathbb{R}^{N\times 3}$ and $\hat{P} \in \mathbb{R}^{N\times 3}$:

$$L_{CD}(\hat{P}, P) = \frac{1}{N}\Big(\sum_{\hat{x_i}\in\hat{P}} \min_{x_j\in P} \|x_i - \hat{x_j}\|_2^2 + \sum_{x_i\in P} \min_{\hat{x_j}\in\hat{P}} \|x_i - \hat{x_j}\|_2^2\Big) \tag{1}$$

where $\hat{P}$ denotes the decoded point set from the embedding, and P denotes the reference point set employed to guide the decoder's learning. For the reference point set, we use the point cloud sampled from the corresponding object CAD models, which are used only in the training process.

Geometric Feature Filtering. Due to the non-Gaussian noise distribution in iPhone LiDAR data (Fig. 5), which should be assumed for most depth camera data, normal estimators might either get perturbed by such noisy features or interpret wrong camera-object rotations. To deal with this sensor-level error, we advocate for the integration of a geometric feature filtering (GFF) module before the modality fusion module. Our approach incorporates the fast Fourier transform (FFT) into the geometric feature encoding. Specifically, the GFF module includes an FFT, a subsequent single layer of MLP, and finally, an inverse-FFT. By leveraging FFT, we can transpose the input sequence of geometric signals to the frequency domain, which selects significant features from noisy input signals. After that, we obtain a more refined geometric embedding that is resilient to the non-Gaussian iPhone LiDAR noise.

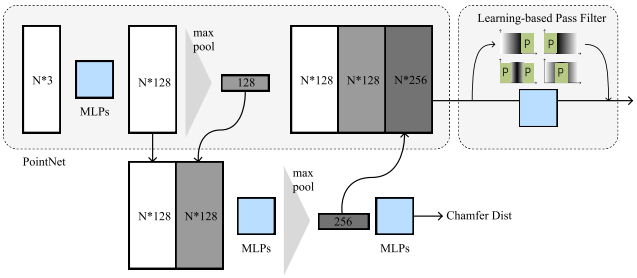

Figure 3. Chamfer distance loss and geometric feature filtering.

2.3. Attention-based RGBD Fusion

Previous papers have emphasized the importance of modality fusion [11, 35] and the benefits of gathering nearest points from the point cloud [11, 27] in RGBD-based pose estimation tasks. While the feature extractor widens each point's receptive field, we aim for features to interact beyond their corresponding points [35] or neighboring points [11]. In predicting the 6DoF pose of a cuboid based on multiple feature descriptors, our focus is on attending to various corner points, rather than solely those in close proximity to each other. To this end, inspired by recent transformer-based models used for modality fusion [4, 8, 9, 18, 22, 28, 31, 36], we leverage the self-attention mechanism [34] to amplify and integrate important features while disregarding the significant LiDAR noise. Specifically, our fusion part is divided into two stages: modality fusion and point-wise fusion (Fig. 2). Both of our fusion modules consist of a standard transformer encoder with linear projection, multi-head attention and layer norm. The former module utilizes the embedding from single-modal encoders and feeds them into a transformer encoder in parallel for cross-modal fusion. The latter fusion module relies on similarity scores among points. It merges all feature embedding in a point-wise manner before feeding them into a transformer encoder. Detailed design and visual analysis of 2 fusion stages are described in our supplemental materials.

2.4. Learning Objective

Based on the overall network structure, our learning objective is to perform 6DoF pose regression, which measures the

disparity between points sampled on the object's model in its ground truth pose and corresponding points on the same model transformed by the predicted pose. Specifically, the pose estimation loss is defined as:

$$(L_{ADD})_{i,p} = \frac{1}{m} \sum_{x \in M} \|(Rx + t) - (\hat{R}_i x + \hat{t}_i)\| \quad (2)$$

where $M \in \mathbb{R}^{m \times 3}$ represents the randomly sampled point set from the object's 3D model, $p = [R|t]$ denotes the ground truth pose, and $\hat{p}_i = [\hat{R}_i|\hat{t}_i]$ denotes the predicted pose generated from the fused feature of the i^{th} point. Our objective is to minimize the sum of the losses for each fusion point, which can be expressed as $L_{ADD} = \frac{1}{N}\sum_i^N (L_{ADD})_{i,p}$, where N is the number of randomly sampled points (token sequence length in the point-wise fusion stage). Meanwhile, we introduce a confidence regularization score (c_i) along with each prediction $\hat{p}_i = [\hat{R}_i|\hat{t}_i]$, which denotes confidence among the predictions for each fusion point:

$$L_{ADD} = \frac{1}{N} \sum_i^N (c_i (L_{ADD})_{i,p} - w log(c_i)) \quad (3)$$

Predictions with low confidence will lead to a low ADD loss, but this will be balanced by a high penalty from the second term with hyper-parameter w. Finally, the Chamfer distance loss, as outlined in Section 2.2, undergoes joint training throughout the training process, leading us to derive our ultimate learning objective as follows:

$$L = L_{ADD} + \lambda L_{CD} \quad (4)$$

where λ denotes the weight of the Chamfer distance loss.

3. Dataset Description

DTTD-Mobile dataset contains 18 rigid objects along with their textured 3D models. The data are generated from 100 scenes, each of which features one or more of the objects in various orientations and occlusion. Following [6], our data generation pipeline is consists of using a professional OptiTrack motion capture system that captures camera pose along the scene and using Apple's ARKit[3] framework to capture RGB images from the iPhone camera and depth information from LiDAR scanner, as illustrated in Fig. 4. After obtaining such data, we then use the open-sourced data annotation pipeline provided by [6] to annotate ground-truth object poses. Through this pipeline, the dataset offers ground-truth labels for 3D object poses and per-pixel semantic segmentation. Additionally, it provides detailed camera specifications, pinhole camera projection matrices, and distortion coefficients. Detailed features and statistics are presented in Table 1. The fact that the DTTD-Mobile dataset includes multiple sets of geometrically similar objects, each having distinct color textures, poses challenges to existing digital-twin localization solutions. To ensure compatibility with other existing datasets, some of the collected objects partially overlap with the YCB-Video [39] and DTTD [6] datasets. Specific details on data acquisition, benchmarking, and evaluation are provided in the appendix.

[3] https://developer.apple.com/documentation/arkit/

4. iPhone LiDAR data analysis

Compared to dedicated depth cameras such as the Microsoft Azure Kinect or Intel Realsense, iPhone 14 Pro LiDAR exhibits more noise and lower resolution at 256×192 depth maps, which leads to high magnitudes of distortion on objects' surfaces. Additionally, it introduces long-tail noise on the projection edges of objects when performing interpolation operations between RGB and depth features. Fig. 5 demonstrates one such example of iPhone 14 Pro's noisy depth data.

To further quantitatively assess the depth noise of each object from the iPhone's LiDAR, we analyze the numerical difference between *LiDAR-measured depth map*, which is acquired directly from iPhone LiDAR, and *reference depth map*, which is derived through ground truth pose annotations. Specifically, to obtain the reference depth map, we leverage ground truth annotated object poses to render the depth projections of each object. We then apply the segmentation mask associated with each object to filter out depth readings that might be compromised due to occlusion. To measure the difference between ground truth and reference depth map, we introduce the *depth-ADD* metric, which calculates the average pixel-wise L1 distance between the ground truth depth map and the reference depth map in each frame. The *depth-ADD* value of each object at frame n is calculated as follows:

$$\text{depth}-\text{ADD}_n = \frac{1}{d} \sum_{i \in D} |\text{depth}_{\text{LiDAR}_i} - \text{depth}_{\text{ref}_i}|, \quad (5)$$

where D denotes the LiDAR depth map and i denotes the index of pixels on it. $\text{depth}_{\text{LiDAR}_i}$ and $\text{depth}_{\text{ref}_i}$ represent the depth values from D and the corresponding depth value from the reference depth map. The set D encompasses all indices i under an object's segmentation mask where both $\text{depth}_{\text{LiDAR}_i}$ and $\text{depth}_{\text{ref}_i}$ yield values greater than zero. The final *depth-ADD* value of each object is the average of such measurements across all N frames:

$$\text{depth}-\text{ADD} = \frac{1}{N} \sum_{n \in N} \text{depth}-\text{ADD}_n \quad (6)$$

Tab. 2 includes the average *depth-ADD* error in each sampled object in the second column. Greater *depth-ADD* values indicate increased distortions and the presence of long-tail noise in the depth data. Our analysis indicates that the

Table 1. Features and statistics of different datasets.

Dataset	Modality	iPhone Camera	Texture	Occlusion	Light variation	# of frames	# of scenes	# of objects	# of annotations
StereoOBJ-1M [24]	RGB	×	✓	✓	✓	393,612	182	18	1,508,327
LINEMOD [13]	RGBD	×	✓	✓	×	18,000	15	15	15,784
YCB-Video [39]	RGBD	×	✓	✓	×	133,936	92	21	613,917
DTTD [6]	RGBD	×	✓	✓	✓	55,691	103	10	136,226
TOD [23]	RGBD	×	✓	×	×	64,000	10	20	64,000
LabelFusion [26]	RGBD	×	✓	✓	✓	352,000	138	12	1,000,000
T-LESS [14]	RGBD	×	×	✓	×	47,762	-	30	47,762
DTTD-Mobile (Ours)	RGBD	✓	✓	✓	✓	47,668	100	18	114,143

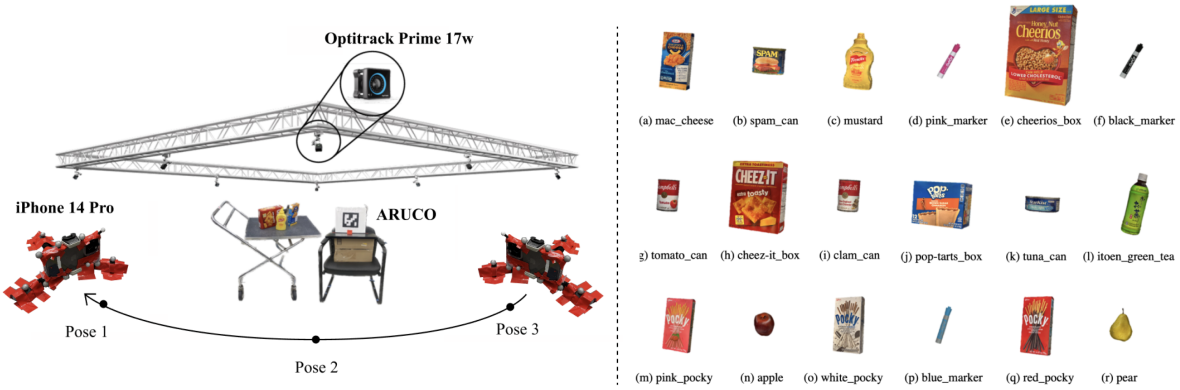

Figure 4. *Left:* Setup of our data acquisition pipeline. *Right:* 3D models of the 18 objects in DTTD-Mobile.

mean *depth-ADD* across all objects is around 0.25m. It is worth noticing that the depth quality varies significantly and could be affected by outliers. For example, there are three objects: *black_marker*, *blue_marker* and *pink_marker* exhibiting greater errors in comparison with the other objects. Detailed depth analysis is reported in the appendix.

5. Experiments

5.1. Evaluation Metrics

We evaluate baselines with the average distance metrics ADD and ADD-S according to previous protocols [6, 39]. Suppose R and t are ground truth rotation and translation and $\tilde{R}$ and $\tilde{t}$ are the predicted counterparts. The ADD metric computes the mean of the pairwise distances between the 3D model points using ground truth pose (R, t) and predicted pose $(\tilde{R}, \tilde{t})$:

$$\text{ADD} = \frac{1}{m} \sum_{x \in M} \|(Rx + t) - (\tilde{R}x + \tilde{t})\|, \qquad (7)$$

where M denotes the point set sampled from the object's 3D model and x denotes the point sampled from M.

The ADD-S metric is designed for symmetric objects when the matching between points could be ambiguous:

$$\text{ADD-S} = \frac{1}{m} \sum_{x_1 \in M} \min_{x_2 \in M} \|(Rx_1 + t) - (\tilde{R}x_2 + \tilde{t})\|. \quad (8)$$

Following previous protocols [20, 24, 25, 35, 39], a 3D pose estimation is deemed accurate if the average distance error falls below a predefined threshold. Two widely-used metrics are employed in our work, namely ADD/ADD-S AUC and ADD/ADD-S(1cm). For commonly used ADD/ADD-S AUC, we calculate the Area Under the Curve (AUC) of the success-threshold curve over different distance thresholds, where the threshold values are normalized between 0 and 1. On the other hand, ADD/ADD-S(1cm) is defined as the percentage of pose error smaller than the 1cm threshold.

5.2. Experimental Results

In this section, we compare the performance of our method DTTDNet with four other 6DoF pose estimators, namely, BundleSDF [37], MegaPose [20], ES6D [27], DenseFusion [35]. While all four methods leverage the benefits of multimodal data from both RGB and depth sources, they differ in the extent to which they emphasize the depth data processing module. Quantitative experimental results are shown in Table 2. Qualitative examples are shown in Fig. 6.

BundleSDF [37] learns multi-view consistent shape and appearance of a 3D object using an object-centric neural signed distance field, leverages frame poses captured in-flight. However, the approach struggles with 3D object reconstruction when faced with large distances, low resolution, or insufficient frame sequence per viewpoint. BundleSDF [37] achieves an ADD AUC of 46.86 and an ADD-S AUC of 55.74. This method failed to reconstruct 8 out of 26 object-scene combinations in DTTD-Mobile. As shown in 6, the failure in 3D object reconstruction results

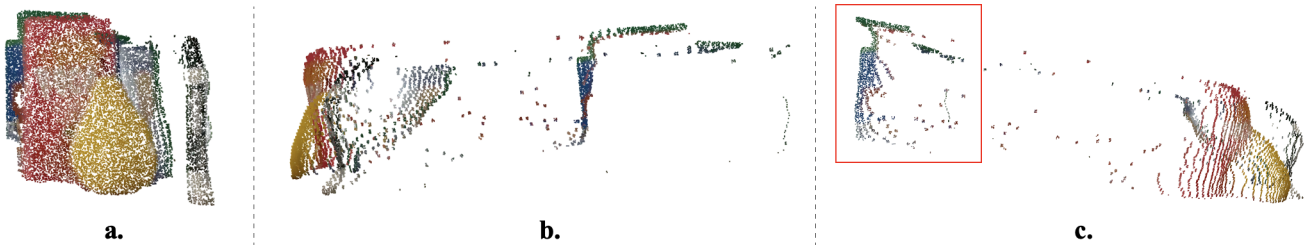

Figure 5. Visualization of an iPhone LiDAR depth scene that shows distortion and long-tail non-Gaussian noise (highlighted inside the red box). (a) Front view. (b) Left view. (c) Right view.

Table 2. **Comparison with diverse 6DoF pose estimation baselines on DTTD-Mobile dataset.** We showcase AUC results of ADD-S and ADD on all 18 objects, higher is better. Based on considered 4 baselines, our model significantly improves the accuracy on most objects. Note that the left-most column indicates the per-object *depth-ADD* error.

	depth-ADD	DenseFusion [35]		MegaPose-RGBD [20]		ES6D [27]		BundleSDF [37]		DTTDNet (Ours)	
Object	Average	ADD AUC	ADD-S AUC	ADD AUC	ADD-S AUC	ADD AUC	ADD-S AUC	ADD AUC	ADD-S AUC	ADD AUC	ADD-S AUC
mac_cheese	0.184	88.10	93.17	78.98	87.94	28.29	57.06	89.95	94.84	**94.06**	**97.02**
tomato_can	0.222	69.10	93.42	68.85	84.48	19.07	56.17	79.62	93.65	**74.23**	**94.01**
tuna_can	0.278	42.90	79.94	8.90	22.11	10.74	26.86	25.05	37.94	**62.98**	**87.05**
cereal_box	0.151	75.20	88.12	59.89	71.53	10.09	53.92	0.00	0.00	**86.55**	**92.74**
clam_can	0.157	**90.49**	96.32	74.11	90.45	17.75	35.92	75.22	96.05	88.15	**96.92**
spam	0.286	53.29	91.14	72.35	86.16	3.17	13.74	**89.24**	**95.12**	52.81	90.83
cheez-it_box	0.152	82.73	92.10	**89.18**	**94.83**	7.81	37.14	42.46	51.69	87.03	93.91
mustard	0.184	78.41	91.31	76.08	85.38	21.89	52.56	84.03	**92.99**	**84.06**	92.15
pop-tarts_box	0.139	82.94	92.58	44.36	58.97	3.44	35.26	82.24	92.01	**84.55**	**92.65**
black_marker	0.769	32.22	38.72	17.38	34.15	2.12	3.72	0.00	0.00	**44.08**	**53.50**
blue_marker	0.370	**66.06**	**74.80**	6.87	12.46	16.88	41.46	0.00	0.00	50.88	61.69
pink_marker	0.410	56.46	67.86	47.84	58.59	1.59	7.01	0.00	0.00	**64.18**	**73.00**
green_tea	0.265	64.37	**93.10**	48.43	70.50	8.80	32.86	60.24	87.29	**64.59**	92.31
apple	0.119	68.97	91.13	32.85	76.43	31.65	58.46	79.27	91.78	**82.45**	**94.80**
pear	0.085	65.66	91.31	35.80	56.73	16.93	32.57	**80.12**	**93.72**	47.83	88.11
pink_pocky	0.231	50.64	67.17	8.69	18.25	0.77	1.93	2.31	6.25	**61.40**	**82.33**
red_pocky	0.245	88.14	93.76	76.49	84.56	25.32	51.16	77.08	91.26	**90.00**	**95.24**
white_pocky	0.265	89.55	94.27	42.83	54.65	17.19	47.45	24.93	26.71	**90.83**	**94.70**
Average	0.239	69.67	85.88	49.02	62.44	13.25	37.38	46.86	55.74	**73.99**	**88.10**

in the absence of object pose tracking in the corresponding scenes. The BundleSDF, as shown in Figure 1, excludes objects in scenes lacking associated pose tracking.

MegaPose [20] employs a coarse-to-fine process for pose estimation. The initial "coarse" module leverages both RGB and depth data to identify the most probable pose hypothesis. Subsequently, a more precise pose inference is achieved through the "render-and-compare" technique. Disregarding the noise in the depth data can also impair the effectiveness of their coarse module, consequently leading to failure in their refinement process. Even with the assistance of a refiner, MegaPose-RGBD [20] only manages to attain an ADD AUC of 49.02 and an ADD-S AUC of 62.44. Its damage and susceptibility to depth noise falls somewhere between DenseFusion [35] and ES6D [27].

DenseFusion [35] treats both modalities equally and lacks a specific design for the depth module, whereas ES6D [27] heavily relies on depth data during training, using grouped primitives to prevent point-pair mismatch. However, due to potential interpolation errors in the depth data, this additional supervision can introduce erroneous signals to the estimator, resulting in inferior performance compared to DenseFusion [35]. DenseFusion [35] achieves 69.67 ADD AUC and 85.88 ADD-S AUC, whereas ES6D [27] only achieves 13.25 ADD AUC and 37.38 ADD-S AUC.

In contrast, our approach harnesses the strengths of both RGB and depth modalities while explicitly designing robust depth feature extraction and selection. In comparison with the above baselines, our method achieves 73.31 ADD AUC and 87.82 ADD-S AUC, surpassing the state of the art with improvements of 1.94 and 3.64 percent in terms of ADD AUC and ADD-S AUC, respectively.

To assess the real-time applicability of DTTDNet, we benchmarked its inference speed on a single NVIDIA RTX A6000 GPU. Our model achieves an average inference time of 0.0172 seconds per object and 0.0378 seconds per frame, corresponding to 58.01 objects per second and 26.43 FPS. These results demonstrate that DTTDNet is well-suited for real-time 6DoF pose estimation, achieving efficient performance without sacrificing robustness or accuracy.

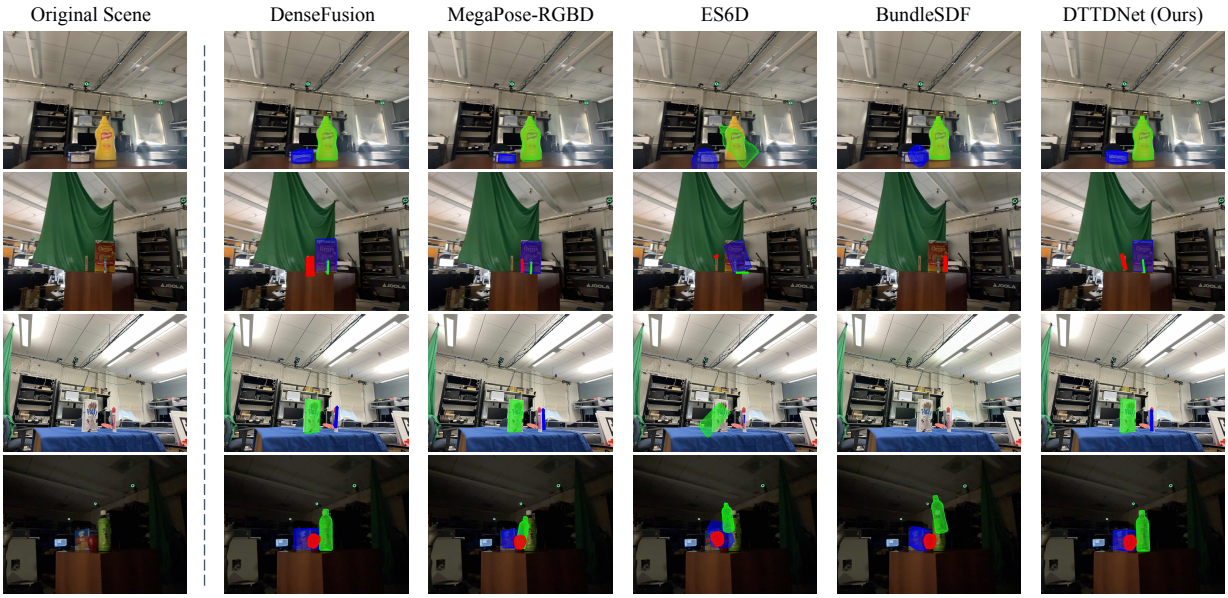

Figure 6. **Qualitative evaluation of different methods.** To further validate our approach, we provide visual evidence of our model's effectiveness in challenging occlusion scenarios and varying lighting conditions, where other models' predictions fail but ours remain reliable. It should be noted that BundleSDF [37] fails to reconstruct 3D objects in some scenes, resulting in the absence of annotations for such objects.

Table 3. **Comparison between 2 datasets with diverse 6DoF pose estimation baselines.** We evaluate the results as the ADD-S AUC on the overlapping objects between the YCB video dataset and the DTTD-Mobile (DTTD-M) dataset, higher is better.

	depth-ADD		DenseFusion [35]		MegaPose-RGBD [20]		ES6D [27]		BundleSDF [37]		DTTDNet (Ours)	
Object	YCB	DTTD-M	YCB	DTTD-M	YCB	DTTD-M	YCB	DTTD-M	YCB	DTTD-M	YCB	DTTD-M
tomato_can	0.011	0.222	93.70	93.42	86.11	84.48	89.02	56.17	68.27	93.65	**96.69**	**94.01**
mustard_bottle	0.005	0.184	95.90	91.31	87.41	85.38	93.13	52.56	**98.21**	**92.99**	97.39	92.15
tuna_can	0.013	0.278	94.90	79.94	91.03	22.11	74.86	26.86	91.11	37.94	**95.78**	**87.05**
average	0.009	0.228	94.83	88.22	88.18	63.99	85.67	45.20	85.86	74.86	**96.62**	**91.07**

5.3. Ablation Studies

In this section, we further delve into a detailed analysis of our own model, highlighting the utility of our depth robustifying module in handling challenging scenarios with significant LiDAR noise.

Evaluation of DTTDNet on other datasets. To show that our proposed pose estimator could also be generalized to other domains, we evaluate our method on the YCB-Video dataset [39], which outperforms MegaPose [20] and ES6D [27] by 6.87 and 7.90 points on ADD(S), respectively, when performing fair comparison (*i.e.*, without any data pre-cleaning and iterative refinement).

We share 3 overlapping objects between the YCB video dataset and our DTTD-Mobile dataset, and demonstrate the models' performance on these objects to examine performance drop of each baseline when shifting the datasets, as shown in Table 3. DTTDNet achieves the highest performance on both datasets, and shows less performance drop when occurring iPhone LiDAR noise. Detailed per-object evaluation is appended in the appendix.

Robustness to LiDAR Depth Error. To answer the question of whether our method exhibits robustness in the presence of significant LiDAR sensor noise when compared to other approaches, we further assess the *depth-ADD* metric, as discussed in Section 4, on DTTDNet versus the four baseline algorithms. Fig. 7 illustrates the correlation between the model performance (ADD) of four methods and the quality of depth information (*depth-ADD*) across various scenes, frames, and 1239 pose prediction outcomes for the 18 objects. Our approach ensures a stable pose prediction performance, even when the depth quality deteriorates, maintaining consistently low levels of ADD error overall.

Effect of Depth Feature Filtering module. Table 4 illustrates the improvement in ADD AUC metrics achieved by our method when integrating geometric feature filtering

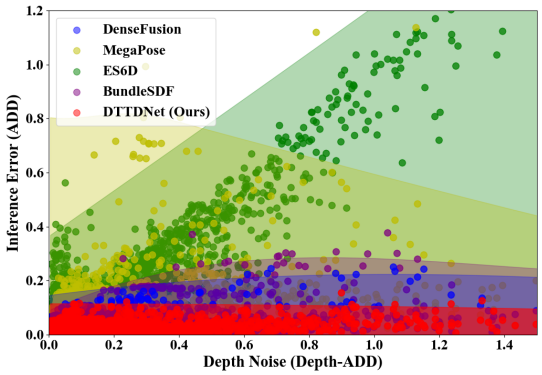

Figure 7. Shadow plot of the relation between the **depth noise** (*depth-ADD*) and the **inference error** (ADD) of considered state-of-the-art methods and proposed DTTDNet.

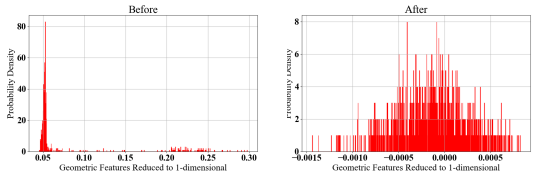

Figure 8. **Probability Distribution of Reduced Geometric Features.** *Left*: before the GFF module. *Right*: after the GFF module.

Table 4. **Effect of Depth Feature Filtering.** M8P4 denotes our model with a fusion stage consisting of 8-layer modality fusion and 4-layer point-wise fusion modules. This table shows the improvement of M8P4 with further incorporation of geometric feature filtering (GFF).

Methods	**ADD AUC**	ADD-S AUC	ADD (1cm)	ADD-S (1cm)
M8P4	72.03	86.44	19.86	**70.50**
+ GFF	**73.31**	**87.82**	**24.35**	66.16

Table 5. **Effect of Object Geometry Augmented CDL.** This table depicts the enhancement in model performance when switching the reference point set from being reliant on the depth map to being augmented by the object model.

Methods	CDL supervised by	**ADD AUC**	ADD-S AUC	ADD (1cm)	ADD-S (1cm)
2*M8P4+GFF	LiDAR depth	73.31	87.82	24.35	66.16
	CAD model	**73.99**	**88.10**	**25.85**	**67.75**

module. To provide a detailed insight into the impact of the GFF module, we conducted principal component analysis (PCA) on both the initial geometric tokens encoded by the PointNet and the filtered version after applying the GFF module, i.e., projected the embedding to a 1-D array with its dominant factor. We visualize the geometric embedding both before and after the application of the GFF module by generating histograms of the dimensionally reduced geometric tokens, as shown in Fig. 8. The distribution of these tokens, as shown in the right subplot, becomes more balanced and uniform after learning-based filtering through the GFF module. The enhanced ADD AUC performance can be attributed to the balanced distribution achieved through the use of the depth robustifying module.

Table 6. **Effect of Layer Number in 2 Fusion Stages.** It shows DTTDNet with different layer number combinations in the fusion stages with one ○ denoting one layer. For all combinations, CDL is used in the geometric feature extraction stage.

Layer Num of #		Metrics			
Modality Fusion	Point-wise Fusion	**ADD AUC**	ADD-S AUC	ADD (1cm)	ADD-S (1cm)
○○	○	70.73	85.42	22.91	67.75
○○	○○	71.37	86.69	15.74	64.44
○○	○○○○	72.06	86.37	19.57	68.37
○○	○	70.73	85.42	22.91	67.75
○○○○	○○	71.76	88.23	20.01	69.03
○○○○○○○○	○○○○	72.03	86.44	19.86	70.50

Effect of Geometry Augmented CDL. We replaced the reference point set for CDL with measured LiDAR depth data as a baseline to demonstrate the effectiveness of our design choice when supervised by the point cloud sampled from 3D object models. In Table 5, we conduct a performance comparison of our approach with these two reference point choices, our design choice achieved higher ADD AUC and ADD-S AUC, as well as higher performance in the more stringent metric, ADD/ADD-S 1cm (Table 5).

Effect of Layer Number Variation in Fusion Stages. Table 6 display the variations brought about by increasing the number of layers at different fusion stages. Overall, adding layer number increases the model's performance in terms of ADD AUC. As we proportionally increase the total number of layers in the modality fusion or point-wise fusion, sustained improvement is observed.

6. Conclusion

We have presented DTTDNet as a novel digital-twin localization algorithm to bridge the performance gap for 3D object tracking in mobile enviroments and with critical requirements of accuracy. At the algorithm level, DTTDNet is a transformer-based 6DoF pose estimator, specifically designed to navigate the complexities introduced by noisy depth data. At the experiment level, we introduced a new RGBD dataset captured using iPhone 14 Pro, expanding our approach to iPhone sensor data. Through extensive experiments and ablation analysis, we have examined the effectiveness of our method in being robust to erroneous depth data. Additionally, our research has brought to light new complexities associated with object tracking in dynamic AR environments.

7. Acknowledgment

This work is supported by the FHL Vive Center at the University of California, Berkeley. We gratefully acknowledge our other team members: Tianjian Xu and Kathy Zhuang for their support with the iPhone capturing app and BundleSDF evaluation on our dataset; Xiang Zhang for his assistance in the further robotic dataset extension[4]; Weiyu Feng and Chenfeng Xu for their valuable discussions and academic insights during the development of this project.

References

[1] Sahar F. Abbas and Fanar M. Abed. Evaluating the accuracy of iphone lidar sensor for building façades conservation. In *Recent Research on Geotechnical Engineering, Remote Sensing, Geophysics and Earthquake Seismology*, pages 141–144, Cham, 2024. Springer Nature Switzerland. 2

[2] Fan Bu, Trinh Le, Xiaoxiao Du, Ram Vasudevan, and Matthew Johnson-Roberson. Pedestrian planar lidar pose (pplp) network for oriented pedestrian detection based on planar lidar and monocular images. *IEEE Robotics and Automation Letters*, 5(2):1626–1633, 2019. 2

[3] Carlos Campos, Richard Elvira, Juan J. G'omez, Jos'e M. M. Montiel, and Juan D. Tard'os. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 1

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 3

[5] Chaojing Duan, Siheng Chen, and Jelena Kovacevic. 3d point cloud denoising via deep neural network based local surface estimation, 2019. 3

[6] Weiyu Feng, Seth Z. Zhao, Chuanyu Pan, Adam Chang, Yichen Chen, Zekun Wang, and Allen Y. Yang. Digital twin tracking dataset (dttd): A new rgb+depth 3d dataset for longer-range object tracking applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3288–3297, 2023. 1, 4, 5

[7] Bo Gu, Jianxun Liu, Huiyuan Xiong, Tongtong Li, and Yuelong Pan. Ecpc-icp: A 6d vehicle pose estimation method by fusing the roadside lidar point cloud and road feature. *Sensors*, 21(10):3489, 2021. 2

[8] Haoming Guo, Seth Z. Zhao, Jiachen Lian, Gopala Anumanchipalli, and Gerald Friedland. Enhancing gan-based vocoders with contrastive learning under data-limited condition. In *2024 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, pages 480–484, 2024. 3

[4]DTTD-Fanuc is an annotated dataset specifically designed for robotic grasping and manipulation with cheaper and low-resolution sensors. It is publicly available in our DTTDNet GitHub repository.

[9] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021. 3

[10] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[11] Yisheng He, Haibin Huang, Haoqiang Fan, Qifeng Chen, and Jian Sun. Ffb6d: A full flow bidirectional fusion network for 6d pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 3

[12] Pedro Hermosilla, Tobias Ritschel, and Timo Ropinski. Total denoising: Unsupervised learning of 3d point cloud cleaning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 52–60, 2019. 3

[13] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Computer Vision – ACCV 2012*, pages 548–562, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 1, 5

[14] Tomáš Hodaň, Pavel Haluza, Štěpán Obdržálek, Jiří Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017. 1, 5

[15] Veli Ilci and Charles Toth. High definition 3d map creation using gnss/imu/lidar sensor integration to support autonomous vehicle navigation. *Sensors*, 20(3):899, 2020. 2

[16] Xiaoke Jiang, Donghai Li, Hao Chen, Ye Zheng, Rui Zhao, and Liwei Wu. Uni6d: A unified cnn framework without projection breakdown for 6d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11174–11184, 2022. 1

[17] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras, 2017. 1

[18] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision, 2021. 3

[19] Maria Kowalska and Janina Zaczek-Peplinska. Evaluation of the lidar in the apple iphone 13 pro for use in inventory work. 2022. 2

[20] Yann Labbé, Lucas Manuelli, Arsalan Mousavian, Stephen Tyree, Stan Birchfield, Jonathan Tremblay, Justin Carpentier, Mathieu Aubry, Dieter Fox, and Josef Sivic. Megapose: 6d pose estimation of novel objects via render compare, 2022. 5, 6, 7

[21] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. *arXiv preprint arXiv:1608.07916*, 2016. 2

[22] Tianjiao Li, Lin Geng Foo, Ping Hu, Xindi Shang, Hossein Rahmani, Zehuan Yuan, and Jun Liu. Token boosting for robust self-supervised visual transformer pre-training, 2023. 3

[23] Xingyu Liu, Rico Jonschkowski, Anelia Angelova, and Kurt Konolige. Keypose: Multi-view 3d labeling and keypoint

estimation for transparent objects. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, 2020. 1, 5

[24] Xingyu Liu, Shun Iwase, and Kris M. Kitani. Stereobj-1m: Large-scale stereo image dataset for 6d object pose estimation. In *ICCV*, 2021. 1, 5

[25] Yuan Liu, Yilin Wen, Sida Peng, Cheng Lin, Xiaoxiao Long, Taku Komura, and Wenping Wang. Gen6d: Generalizable model-free 6-dof object pose estimation from rgb images, 2023. 5

[26] Pat Marion, Peter R Florence, Lucas Manuelli, and Russ Tedrake. Label fusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3325–3242. IEEE, 2018. 1, 5

[27] Ningkai Mo, Wanshui Gan, Naoto Yokoya, and Shifeng Chen. Es6d: A computation efficient and symmetry-aware 6d pose regression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6718–6727, 2022. 1, 2, 3, 5, 6, 7

[28] Sayak Paul and Pin-Yu Chen. Vision transformers are robust learners, 2021. 3

[29] Ivan Permozer and Tihomir Orehovački. Utilizing apple's arkit 2.0 for augmented reality application development. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1629–1634, 2019. 1, 2

[30] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. 3

[31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 3

[32] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 134–144, 2019. 1

[33] Jiaming Sun, Zihao Wang, Siyu Zhang, Xingyi He, Hongcheng Zhao, Guofeng Zhang, and Xiaowei Zhou. OnePose: One-shot object pose estimation without CAD models. *CVPR*, 2022. 2

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 3

[35] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. 2019. 1, 2, 3, 5, 6, 7

[36] Kehan Wang, Seth Z. Zhao, David Chan, Avideh Zakhor, and John Canny. Multimodal semantic mismatch detection in social media posts. In *Proceedings of IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*, 2022. 3

[37] Bowen Wen, Jonathan Tremblay, Valts Blukis, Stephen Tyree, Thomas Muller, Alex Evans, Dieter Fox, Jan Kautz, and Stan Birchfield. Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects. *CVPR*, 2023. 5, 6, 7

[38] Xinshuo Weng and Kris Kitani. Monocular 3d object detection with pseudo-lidar point cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[39] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. 2018. 1, 4, 5, 7

[40] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. *arXiv preprint arXiv:1906.06310*, 2019. 2

[41] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network, 2017. 2