Prior-free 3D Object Tracking

Xiuqiang Song[1,3] Li Jin[1,3] Zhengxian Zhang[1,3] Jiachen Li[4]
Fan Zhong[2,3] Guofeng Zhang[5] Xueying Qin[1,3*]
[1]School of Software, Shandong University
[2]School of Computer Science and Technology, Shandong University
[3]Engineering Research Center of Digital Media Technology, Ministry of Education, P.R. China
[4]Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center, Qilu University of Technology
[5]State Key Lab of CAD&CG, Zhejiang University
song_xq, jinli, zhangzhengxian@mail.sdu.edu.cn jcli@qlu.edu.cn
zhongfan@sdu.edu.cn zhangguofeng@zju.edu.cn qxy@sdu.edu.cn

Abstract

In this paper, we introduce a novel, truly prior-free 3D object tracking method that operates without given any model or training priors. Unlike existing methods that typically require pre-defined 3D models or specific training datasets as priors, which limit their applicability, our method is free from these constraints. Our method consists of a geometry generation module and a pose optimization module. Its core idea is to enable these two modules to automatically and iteratively enhance each other, thereby gradually building all the necessary information for the tracking task. We thus call the method as Bidirectional Iterative Tracking(BIT). The geometry generation module starts without priors and gradually generates high-precision mesh models for tracking, while the pose optimization module generates additional data during object tracking to further refine the generated models. Moreover, the generated 3D models can be stored and easily reused, allowing for seamless integration into various other tracking systems, not just our methods. Experimental results demonstrate that BIT outperforms many existing methods, even those that extensively utilize prior knowledge, while BIT does not rely on such information. Additionally, the generated 3D models deliver results comparable to actual 3D models, highlighting their superior and innovative qualities. The code is available at https://github.com/songxiuqiang/BIT.git.

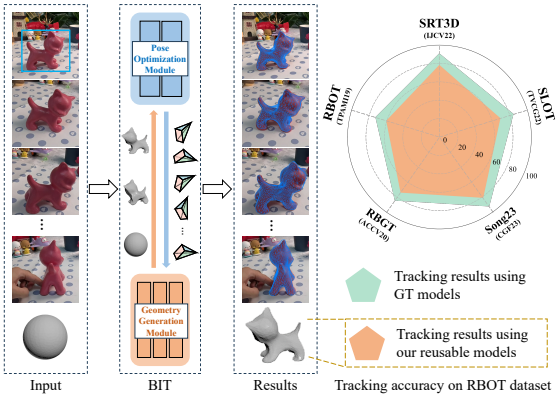


Figure 1. We propose a prior-free 3D object tracking method called BIT, which is both model-free and training-free. The pose optimization and geometry generation modules iteratively enhance each other to ultimately realize object tracking. The generated model can be reused, achieving excellent results.

1. Introduction

The advancement of embodied intelligence technology necessitates a deep understanding of the real world, and a crucial foundation for this understanding is the accurate and continuous real-time 6DoF pose estimation of objects in the real world, also known as 3D object tracking. Existing methods typically rely on predefined 3D model priors [4, 20, 22] or training priors [9, 30], or a combination of both [27], to achieve accurate tracking results. While the use of priors simplifies the tracking task, it also incurs high training costs and the need for precise model reconstruction, which significantly limits the flexibility and scalability of these methods across diverse environments and object

*Xueying Qin is the corresponding author.

types. In this paper, we explore, for the first time, a truly prior-free 3D object tracking paradigm that is both model-free and training-free, specifically without requiring given 3D models or pose-annotated training images.

We introduce a Bidirectional Iterative Tracking (BIT) to tackle the challenging task of prior-free tracking. The key idea of BIT is to automatically generate the necessary information for tracking, rather than relying on pre-existing priors that serve a similar purpose. BIT relies solely on monocular RGB data as input. It comprises two key modules: the geometry generation module and the pose optimization module. The geometry generation module can automatically deform a sphere to match the target object's shape, while the pose optimization module tracks the pose and selects the appropriate information as a foundation for geometry generation. These modules work in cooperation, iteratively enhancing each other's functionality. Additionally, the information generated by BIT can be used immediately or stored for future reuse, allowing other tracking systems to seamlessly integrate it. The core components of BIT include a tracker SLOT [4], a segmenter SAM [6], and a differentiable renderer SoftRas [8].

The BIT method improves upon previous methods by completely eliminating the reliance on model priors and training priors (the training of SAM is not specific to the tracked object, and in our method it also can be replaced with other segmentation method). Existing 3D object tracking methods typically depend on provided priors to reduce the difficulty of tracking, which can be classified into three main categories: *Model-based methods*, which rely on CAD models or reconstructed models as priors and cannot function without them. *Training-based methods*, which involve training on specific datasets and then generalizing to unseen objects. *Model and training-based methods*, which rely on both models and training data simultaneously. In practical scenarios, acquiring such models and training data incurs significant costs, which clearly limits the flexibility and applicability of these methods. BIT overcomes these limitations by functioning independently of such priors, making it more versatile and cost-effective. The key contributions of this paper are as follows:

- The BIT framework is proposed, which realizes prior-free tracking of 3D objects, encompassing both model-free and training-free.
- An effective bidirectional iterative tracking method is proposed, where the pose optimization module and the geometry generation module can iteratively enhance each other, achieving synchronized improvement in their effects and jointly completing the tracking task.
- The 3D models automatically generated by BIT can be easily reused and seamlessly integrated into many other tracking systems without additional processing, providing convenience for the utilization of other tracking methods.

2. Related Works

Based on our knowledge, there are currently no 3D object tracking methods that do not rely on either model priors or training priors. In this section, we will first provide an overview of 3D object tracking and methods for single-frame pose estimation relevant to our work, followed by an introduction to differentiable rendering. Single-frame pose estimation determines an object's pose from a single frame without relying on previous frames, leading to higher computational demands due to a larger search space. In contrast, 3D object tracking uses pose information from previous frames for a more efficient and accurate estimation, optimizing within a smaller search space.

2.1. Model-based Methods

Precise 3D models can provide strong constraints for tracking, which is particularly important for textureless or weakly textured objects. In recent years, many model-based methods [4, 17, 19, 20, 22, 23] have achieved excellent tracking results by utilizing well-designed energy functions. These energy functions are typically constructed from interpretable features, such as the color distribution of pixels [4, 17, 19, 20, 23] or edges [3, 22, 24, 25] in the frame. These methods do not rely on deep networks, thereby avoiding issues related to training and generalization. They typically offer good tracking accuracy and real-time performance while having low hardware requirements. When precise CAD models are available, these methods are generally the best choice for object tracking. However, obtaining precise CAD models is not always feasible, which is the main limitation of these methods.

2.2. Training-based Methods

Some methods do not rely on explicit 3D models as priors but require training on specific datasets to acquire the necessary knowledge, subsequently using this learned information to predict poses. Additionally, some methods require a specific number of reference frames and corresponding camera poses to reconstruct an object's point cloud model via Structure-from-Motion (SfM), as seen in OnePose [21] and OnePose++ [2]. After reconstruction, the object's pose is estimated by matching the 2D image with the 3D point cloud. BundleSDF [29] utilizes RGB-D data for 3D reconstruction while tracking. Some methods, like Gen6D [9] and MFOS [7], estimate the pose directly from given reference frames. These techniques typically depend on a series of provided reference frames, thereby introducing certain constraints. FoundationPose [30] trains neural networks on large datasets to acquire as much knowledge as possible, enabling them to handle a wider range of scenarios. Overall, while these methods reduce dependence on model priors to some extent, their reliance on training can lead to generalization issues, and the collection of training data are costly.

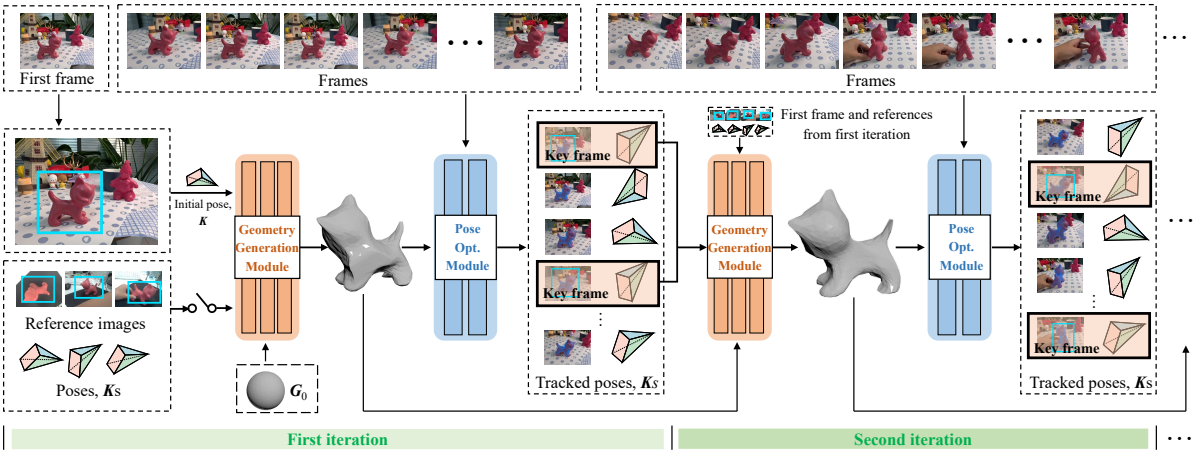


Figure 2. Overview of our method BIT. Our method starts with the first frame, using a bounding box to identify the object to track, optionally incorporating available reference frames. The geometry generation module quickly inverse renders a mesh model for the pose optimization module, which in return supplies additional key frames as reference frames. This mutual enhancement between the modules transforms a model-free scene into a model-based scene, leading to improved results.

2.3. Model and Training-based Methods

Some methods rely on both model priors and training priors to achieve better tracking accuracy or address specific problems (such as category-level pose estimation [26, 28, 31]). DeepAC [27] combines deep networks with precise CAD models to achieve enhanced results. Additionally, some methods [26, 28, 31] can perform tracking with imprecise models. These methods estimate the poses of different instances within a category based on a shared category prior (which can be considered a model prior, typically 3D point clouds). However, these methods usually heavily rely on depth information and struggle to achieve ideal results when relying solely on RGB inputs. Another significant issue is that these methods require extensive training and are prone to generalization challenges.

2.4. Differentiable Rendering

Traditional rendering pipelines are non-differentiable due to discrete sampling in rasterization. Differentiable rendering transforms this into a differentiable process, allowing gradients to flow from 2D images to 3D models, which provide a basis for generating 3D geometry from 2D data. Some methods [5, 10] use handcrafted functions for gradient approximation, while SoftRas [8] achieves full differentiability with an aggregation function. Differentiable rendering is applied in reconstruction, shape fitting, and technologies like NeRF [11]. It requires object poses and segmented contours or masks, often needing multiple views for better results. A method [15] combines 2D trackers with differentiable rendering to reconstruct the geometry, color, texture, and 6DoF pose of 3D objects. However, this method is time-consuming, requiring about 2 seconds per frame, whereas we address a real-time, prior-free problem.

3. Method

The overview of our method BIT is shown in Fig. 2. BIT takes RGB images as input, with the first frame utilizing a bounding box to indicate the target object for tracking. Additionally, if available, it can incorporate a limited number of reference frames. Our method is capable of simultaneously performing iterative real-time tracking and model generation. It primarily consists of two mutually enhancing components: the geometry generation module and the pose optimization module.

After receiving the first frame and the reference frames (if available), the geometry generation module can quickly generate an initial model by inverse rendering using the given data and send the generated model to the pose optimization module. The pose optimization module then employs this generated model for continuous tracking, selecting key frames from the tracking results and feeding this data back to the geometry generation module. The geometry generation module, in turn, utilizes this feedback to further optimize the generated model. These two processes iterate continuously, simultaneously optimizing both the generated model and tracking performance. Once the generated model reaches a sufficient level of precision, only the pose optimization module needs to be executed.

Throughout the entire process, the pose optimization module does not require any training. The reference for inverse rendering used by the geometry generation module is automatically generated during the tracking process, also eliminating the need for training.

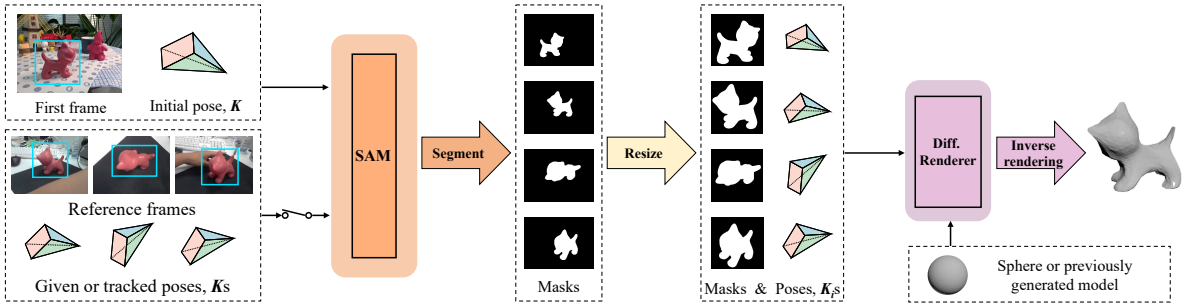


Figure 3. The geometry generation module. This module takes the first frame, along with a bounding box, an initial pose, and camera intrinsic $\boldsymbol{K}$ as input. It can also incorporate additional reference frames if available. A better mesh model can be obtained by inverse rendering a sphere or the previously generated model.

3.1. Geometry Generation Module

The geometry generation module rapidly generates an object's mesh model by iteratively inverse rendering during the tracking process. The generated model is then supplied to the pose optimization module. Figure 3 illustrates the overview of the geometry generation module.

The geometry generation module takes as input the first frame $\boldsymbol{\mathcal{I}}_0$, an initial pose $\boldsymbol{T}_0 = [\boldsymbol{R}_0|\boldsymbol{t}_0]$, and a bounding box $\boldsymbol{b}_0$ that specifies the target object. Here, $\boldsymbol{R}_0 \in \mathbb{R}^{3\times3}$ represents the rotation matrix, and $\boldsymbol{t}_0 \in \mathbb{R}^3$ is the translation vector. Additionally, the camera intrinsic matrix $\boldsymbol{K} \in \mathbb{R}^{3\times3}$ is also provided.

Additionally, if available, the geometry generation module can incorporate reference frames $\{\boldsymbol{\mathcal{I}}_i\}$, corresponding poses $\{\boldsymbol{T}_i\}$, and bounding boxes $\{\boldsymbol{b}_i\}$. Since the post-processing for the initial frame and reference frames is identical, we assume the presence of reference frames to streamline the description of our method. Furthermore, the pose optimization module can automatically generate these reference frames during the iterative process. For the sake of simplicity, we will use $\{\boldsymbol{\mathcal{I}}_i\}$ to refer to both $\boldsymbol{\mathcal{I}}_0$ and $\{\boldsymbol{\mathcal{I}}_i\}$, and apply the same convention to $\{\boldsymbol{T}_i\}$ and $\{\boldsymbol{b}_i\}$.

The input frames $\{\boldsymbol{\mathcal{I}}_i\}$ are first fed to a segmenter SAM [6] to obtain the object's masks, with the bounding boxes $\{\boldsymbol{b}_i\}$ serving as prompts. To ensure the object area defined by $\{\boldsymbol{b}_i\}$ is not too small and to facilitate easier optimization during post-processing, we crop the object area and resize it to a fixed size, resulting in the resized masks $\{\boldsymbol{\mathcal{M}}_i\}$. During this process, we adjust the camera intrinsics from $\boldsymbol{K}$ to $\boldsymbol{K}_i$ for each resized mask $\boldsymbol{\mathcal{M}}_i$. This adjustment is necessary because $\boldsymbol{K}$ influences the size and position of the image, ensuring that the input pose $\{\boldsymbol{T}_i\}$ remains compatible with the modified masks $\boldsymbol{\mathcal{M}}_i$. For further details on this process, please refer to our supplementary materials.

Starting with a 3D sphere $\boldsymbol{G}_0$ as the initial model and following the SoftRas [8] method, we iteratively deform this sphere $\boldsymbol{G}_0$ into the target mesh model $\boldsymbol{G}^*$ using the resized masks $\{\boldsymbol{\mathcal{M}}_i\}$ and poses $\{\boldsymbol{T}_i\}$. SoftRas is chosen due to its rapid processing speed. Since the masks provide comprehensive cues and supervision for inverse rendering, there is no need to train the differentiable renderer. We simplified the loss function for mesh inverse rendering in SoftRas, resulting in a very concise loss function $\mathcal{L}$:

$$\mathcal{L} = \mathcal{L}_{IoU}(\{\boldsymbol{G}, \boldsymbol{T}_i, \boldsymbol{K}_i\}, \{\boldsymbol{\mathcal{M}}_i\}) + \mu\mathcal{L}_l(\boldsymbol{G}), \quad (1)$$

where $\mathcal{L}_{IoU}$ represents the 2D IoU loss between the given masks $\{\boldsymbol{\mathcal{M}}_i\}$ and the projected masks of the model $\boldsymbol{G}$ under poses $\{\boldsymbol{T}_i\}$ and intrinsics $\{\boldsymbol{K}_i\}$, $\boldsymbol{G}$ represents the intermediate 3D model transitioning from $\boldsymbol{G}_0$ to $\boldsymbol{G}^*$. $\mathcal{L}_l$ is the Laplacian regularization term used to control the smoothness of the geometry. The parameter μ acts as a balancing factor. The loss function is intentionally kept simple to facilitate the rapid convergence of the input geometry to a specific shape suitable for tracking. Additionally, a straightforward loss function is crucial for generalization. For more details please see our supplementary materials.

Throughout the process, the 3D model initially starts as a sphere, and each iteration generates a mesh model that can be utilized by the pose optimization module. Except for the first iteration, each subsequent iteration receives a fixed number of key frames generated by the pose optimization module as inputs. Additionally, in every iteration, the first frame and reference frames (if available) are consistently used as inputs. Since the contours of the masks segmented by SAM are not smooth and have minor pixel discrepancies, we found that the inverse-rendered model might appear slightly smaller than the actual object. To compensate for this, we scale the inverse-rendered model by a factor of 1.05, which is an empirically determined parameter.

3.2. Pose Optimization Module

The pose optimization module is shown in Fig. 4, which has two main purposes: First, it receives the generated 3D mesh model and uses it to quickly estimate the object's pose in the frame. Second, it generates key frames and the object's bounding boxes, and sends these data along with their corresponding poses to the geometry generation module.

Method	Input	Training	Model	Refers	black_drill	duplo_dude	rinse_aid	toy_plane	vim_mug	Avg.↑
PoseRBPF [1]†	RGB-D*	✓	✓	✘	75.3	84.1	87.2	48.6	59.0	70.8
LatentFusion [13]†	RGB-D	✓	✘	✓	**89.4**	88.9	67.4	**82.9**	40.0	73.7
LatentFusion [13]‡	RGB-D	✓	✘	✓	74.1	75.4	63.0	55.0	38.3	61.1
PVNet [14]‡	RGB	✓	✓	✘	49.5	43.3	72.9	48.6	67.9	56.4
Gen6D [9]‡	RGB	✓	✘	✓	64.9	59.2	72.0	69.8	51.0	63.4
Ours(0)	RGB	✘	✘	✘	63.1	69.7	80.5	76.2	81.4	74.2
Ours(3)	RGB	✘	✘	✓	82.9	**89.9**	**88.3**	81.7	**88.4**	**86.2**

Table 1. Tracking results on the MOPED dataset under the ADD-AUC (0–10 cm). Ours(0) uses no reference frames, Ours(3) uses 3 reference frames. Bold indicates the best result, and underline indicates the second best. † denotes the data reported by LatentFusion, and ‡ denotes the results reported by Gen6D under its configuration. *According to the description in LatentFusion's experiments, we infer that this method uses RGB-D data.

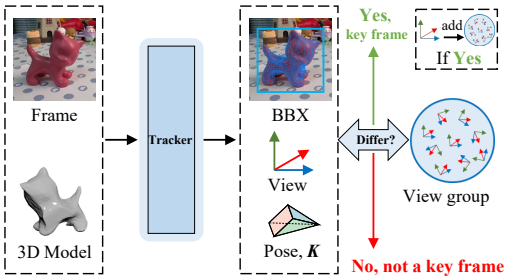


Figure 4. The pose optimization module, which can track the object's pose and generate key frames for model generation module.

We utilize the tracking method SLOT [4] as the baseline for tracking the object's pose. This method accepts the generated model as input and then optimizes the object's pose using the color features in the input RGB image around the projected mask of the model. Since it does not include any deep networks, it does not require training. For a detailed explanation of this method, please refer to our supplementary materials. Next, we will provide a detailed explanation of how to generate the key frames and bounding boxes required for the geometry generation module.

The key frames should encompass as many viewpoints as possible to enhance the effectiveness of the geometry generation module. Concurrently, to reduce computational load, redundant viewpoints should be minimized. To achieve this, we construct a view group, denoted as $\boldsymbol{\mathcal{V}}$, to store the viewpoints of frames calculated from the tracked poses:

$$\boldsymbol{\mathcal{V}} = \{\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_{n-1}\}, \tag{2}$$

where $\boldsymbol{v}_i \in \mathbb{R}^3$ is the $i-$th view, which can be calculated from the corresponding pose $\boldsymbol{T}_i = [\boldsymbol{R}_i|\boldsymbol{t}_i]$:

$$\boldsymbol{v}_i = \boldsymbol{R}_i^\top \boldsymbol{t}_i / ||\boldsymbol{t}_i||. \tag{3}$$

When a new frame $\boldsymbol{\mathcal{I}}_n$ is tracked and the camera view $\boldsymbol{v}_n$ is established, we compute the angular differences between $\boldsymbol{v}_n$ and all the views in the set $\boldsymbol{\mathcal{V}}$ as follows:

$$\Delta \boldsymbol{a}_i = acos(\frac{\boldsymbol{v}_n \cdot \boldsymbol{v}_i}{||\boldsymbol{v}_n|| ||\boldsymbol{v}_i||}), for\ \boldsymbol{v}_i \in \boldsymbol{\mathcal{V}}. \tag{4}$$

Method	Refers	Key Frames	Iters	Model Acc. ↓
Ours(3)	3	3	3	0.035
Ours(6)	6	12	3	0.011
Ours(12)	12	12	3	**0.010**

Table 2. Our generated models in different cases. Model accuracy is measured by the normalized Hausdorff distance.

If all the $\Delta \boldsymbol{a}_i$ values exceed a certain threshold $\boldsymbol{a}_{\text{min}}$, we consider the frame $\boldsymbol{\mathcal{I}}_n$ to be a key frame candidate.

We then project the 3D mesh model according to the tracked pose $\boldsymbol{T}_n$ to obtain a bounding box $\boldsymbol{b}_n$. If $\boldsymbol{b}_n$ is more than 10 pixels away from the image border, we consider $\boldsymbol{b}_n$ to be valid and designate $\boldsymbol{\mathcal{I}}_n$ as a true key frame, subsequently adding $\boldsymbol{v}_n$ to $\boldsymbol{\mathcal{V}}$. If a sufficient number of key frames are selected, these frames, along with their corresponding bounding boxes and poses, will be sent to the geometry generation module to perform inverse rendering.

4. Experiments

We conducted experiments on both publicly available datasets and real-world scenarios, presenting both quantitative and qualitative results. The experimental setup consisted of a desktop computer equipped with an Intel 12700 CPU and a single NVIDIA GTX 3080 (10GB) GPU. In our experiments, we set $\boldsymbol{a}_{\text{min}} = 3°$. The masks were resized to 256×256 resolutions during the inverse rendering process. We employed the Adam optimizer, configured with a learning rate of 0.01 and beta values of (0.50, 0.99), to optimize the model. The initial model is a 3D sphere with 1302 vertices. Additionally, $\mu = 0.1$ was set in the loss function. Each inverse rendering iteration during the model generation process consisted of 200 optimization steps.

In cases where we used reference frames on the dataset, we employed an algorithm that utilizes the golden ratio to evenly distribute a certain number of viewpoints on the surface of a sphere and then selected the frames closest to these generated viewpoints to serve as reference frames.

(a) Tracking results and generated models in iterations without reference frame

(b) Tracking results and generated models in iterations with 3 reference frames

Figure 5. Tracking results and generated models in 3 iterations on the MOPED dataset, displayed from left to right.

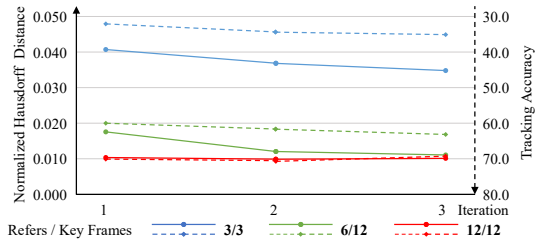


Figure 6. The accuracy of the generated models (solid lines) and tracking (dashed lines) across iterations (note the axis direction).

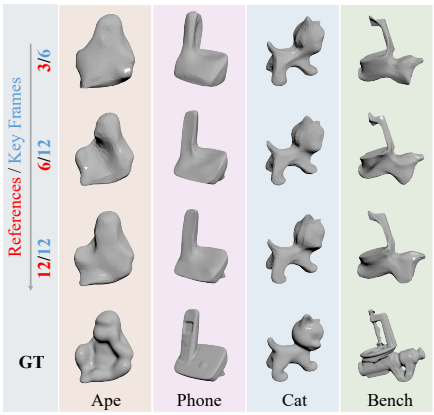


Figure 7. The final generated models (3 iterations) in different reference frames and key frames on the RBOT dataset.

4.1. Results on the MOPED Dataset

The MOPED dataset [13] is a real-world captured dataset containing 11 objects, each comprising several RGB-D video sequences divided into reference and evaluation sets, with a resolution of 640×480. The poses are obtained through a combination of multiple methods [12, 16, 32, 33]. Due to the annotation manner, the poses in some sequences may not be entirely accurate. Therefore, following the approach of Gen6D [9], we selected 5 relatively reliable objects for evaluation. For each object, there are 100-600 reference images and 100-300 query images.

We use the ADD-AUC (0-10 cm) metric to evaluate the tracking results. We test our method in two versions: Ours(0) uses no model or training priors and no reference frames, generating 3 key frames per iteration. Ours(3) also uses no model or training priors but employs 3 reference frames, generating 6 key frames per iteration. For each object, we perform model generation for up to 3 iterations.

Since the proposed method addresses a new problem, to the best of our knowledge, there are currently no directly comparable methods. Therefore, we selected the methods most similar to ours for comparison. We compare our results with PoseRBPF [1], LatentFusion [13], PVNet [14], and Gen6D [9]. The results are shown in the Tab. 1. We also present a part of tracking results, as well as the generated models, as shown in Fig. 5.

Even without any priors and reference frames, our method (Ours(0)) outperforms all other methods in average. With 3 reference frames, our method (Ours(3)) significantly outperforms the others. Notably, our method requires no training and uses only RGB data, whereas other methods were trained on the tested or other objects. Additionally, compared to other methods, our method demonstrates more uniform accuracy across various objects, indicating that it is data-independent and a truly training-free approach.

4.2. Results on the RBOT Dataset

During tracking, our method can rapidly generate the 3D model of an object, by comparing the performance of various trackers using both precise ground truth CAD models and our generated models on the RBOT dataset [23], we explored the models' reusability.

The RBOT dataset is a 3D object tracking dataset with 18 objects and 72 sequences, each containing 1000 test frames, with a resolution of 640×512, covering a baseline regular scenario and dynamic light, noisy, and occlusion scenarios. It has been used to evaluate some advanced 3D track-

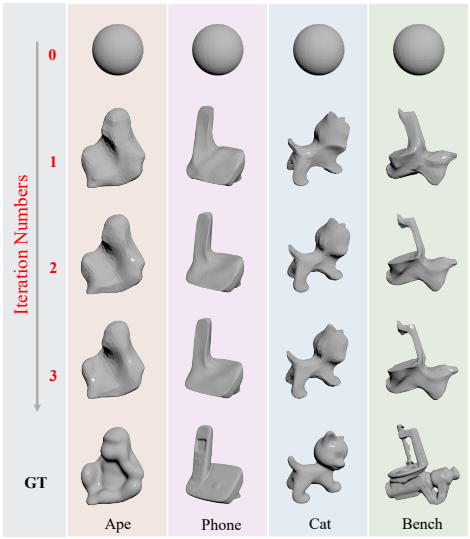


Figure 8. Generated models in the case of 6 reference frames and 12 key frames on the RBOT dataset, displayed from top to bottom in each iteration.

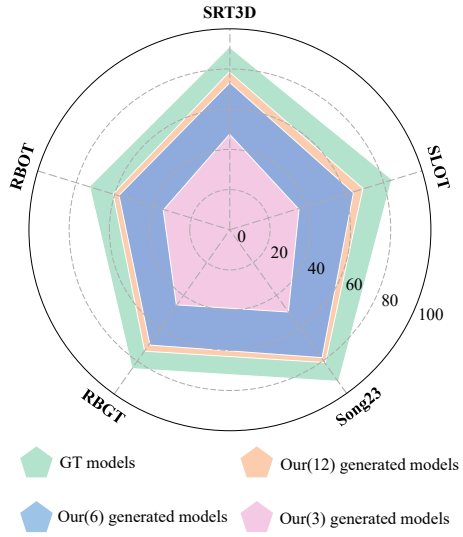


Figure 9. Tracking accuracy of various trackers on the RBOT dataset using GT models and the our generated models.

Method	Model	Reg.	Dyn.	Noisy	Occ.	Avg.↑
RBOT [23]	GT	79.9	81.2	56.6	73.3	72.8
	Ours(3)	38.1	37.5	25.9	32.4	33.5
	Ours(6)	63.2	62.9	42.3	56.1	56.1
	Ours(12)	**67.8**	**67.0**	**44.9**	**60.2**	**60.0**
SLOT [4]	GT	89.9	90.7	69.6	88.9	84.8
	Ours(3)	38.8	38.4	26.9	36.1	35.1
	Ours(6)	68.8	68.2	49.8	66.0	63.2
	Ours(12)	**75.2**	**74.3**	**53.3**	**72.4**	**68.8**
Song23 [17]	GT	95.4	94.9	86.2	93.2	92.4
	Ours(3)	52.7	50.9	44.8	48.3	49.2
	Ours(6)	81.5	79.8	70.9	77.7	77.5
	Ours(12)	**85.7**	**83.8**	**74.6**	**81.9**	**81.5**
RBGT [18]	GT	90.0	90.6	71.5	85.6	84.4
	Ours(3)	48.9	47.8	37.9	44.3	44.7
	Ours(6)	73.9	73.3	57.6	68.6	68.3
	Ours(12)	**79.7**	**78.9**	**61.4**	**74.0**	**73.5**
SRT3D [19]	GT	94.2	94.6	81.7	93.2	90.9
	Ours(3)	49.7	48.9	40.9	46.4	46.5
	Ours(6)	75.9	75.2	63.8	72.7	71.9
	Ours(12)	**81.8**	**80.9**	**69.1**	**79.4**	**77.8**

Table 3. Tracking accuracy with GT models and the generated models by our method on the RBOT dataset. Bold indicates the best result except for the ground truth.

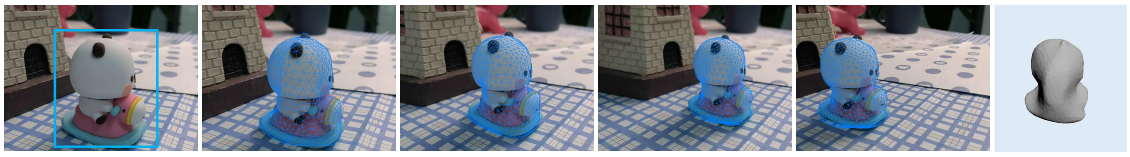
ing methods, such as SRT3D [19], RBGT [18], SLOT [4], Song23 [17] and the RBOT method [23].

Due to the significant differences between frames in the RBOT dataset and the rapid movement of objects, we used 3-12 reference frames, and then tracked the first 100 frames in the regular scenario, and generate the objects' models during tracking. For each object, we performed up to 3 iterations. Then these generated models served as inputs for other model-based trackers for testing across all scenes of the entire dataset, allowing us to compare their performance against the ground truth CAD models.

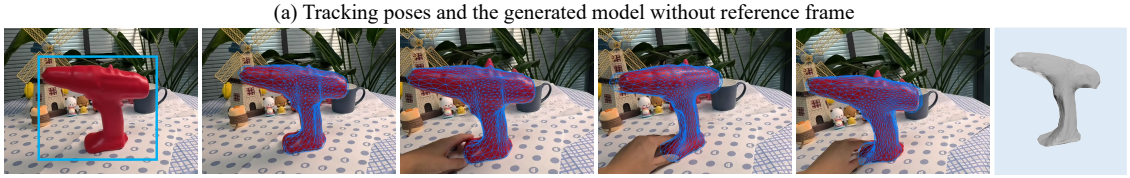
We employed the $5\ cm/5°$ metric to measure the tracking accuracy, which indicates success in a frame if the translation error is less than $5\ cm$ and the rotation error is less than $5°$. Otherwise, it is considered a failure, and the pose is reset to the ground truth. The proportion of successful frames represents the tracking accuracy. We use the normalized Hausdorff distance between the generated models and ground truth models to measure the models accuracy.

Table 2 shows the models' accuracy (in normalized Hausdorff distance) in the last iteration, and Fig. 6 shows the normalized Hausdorff distance in every iteration. It can be seen that the generated models have relatively high accuracy with 6 or more reference frames. Additionally, by the second iteration, the model is already close to the final one. This indicates the proposed method's rapid convergence. Figures 7 and 8 also display portions of the generated models, demonstrating the effectiveness of our method. Furthermore, Figure 6 illustrates the variation in tracking accuracy of the baseline tracker [4] across the entire dataset using the intermediate generated models. The results show that the tracking accuracy improves as the models become more accurate.

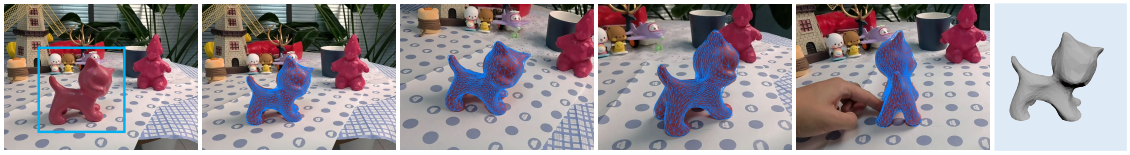
The test results of the 5 tracking methods using the final generated models on the RBOT dataset are presented in Tab. 3 and Fig. 9. It is clear from these results that when given 6 or more reference frames, the generated models can achieve results very close to the ground truth CAD models. This proves the effectiveness of our method, as it can



(a) Tracking poses and the generated model without reference frame



(b) Tracking poses and the generated model with 3 references frames



(c) Tracking poses and the generated model with 6 references frames

Figure 10. Qualitative results on the real world scenes. From left to right, the first frame, tracking results, and the generated models.

effectively transform a model-free tracking problem into a model-based tracking problem.

4.3. Qualitative Results

We captured RGB images of real-world scenes to test the proposed method, with a resolution of 640×480. We evaluated our method under 3 conditions: (1) no reference and 3 key frames, (2) 3 reference frames and 3 key frames, and (3) 6 reference frames and 6 key frames. The reference frames are from the RBOT datatset. For each object, we performed up to 3 iterations. The tracking results are shown in Fig. 10. Under all 3 conditions, our method successfully achieved effective tracking and generated object models. Further experimental results can be found in our supplementary video.

4.4. Time Analysis

With a resolution of 640×480, tracking a frame takes about 30 ms. When using 6 reference frames and 12 key frames for generating models, each iteration takes about 10 seconds. Therefore, a 3D model of an object can be obtained in about 30 seconds over 3 iterations. Notably, our method only utilizes a single NVIDIA 3080 (10GB) GPU. After completing the model generation, only tracking is required, allowing our method become real-time (30 FPS). Additionally, tracking can be run as a background thread to reduce time consumption.

5. Limitations

Our method has several limitations that need to be addressed. Firstly, it is incapable of generating recessed features on object surfaces and cannot accurately model specific topologies, such as ring-shaped or hollow structures. Secondly, in challenging scenarios, SAM's segmentation performance may be affected, particularly in cases involving occlusion, motion blur, or similar colors, and the object's complex geometry can also have a certain impact; errors in the mask may degrade the initially rendered model and tracking accuracy, further complicating iterative improvements. Lastly, tracking failures may occur in challenging scenes, particularly with fast-moving objects and when there is a lack of reference frames.

6. Conclusions

We have proposed a prior-free tracking method based on RGB data, capable of quickly generating 3D models of objects while tracking them. This approach transforms a model-free tracking problem into a model-based one without requiring pose-annotated training. The method operates when the target object is specified in the first frame and can incorporate reference frames if available. The generated models can be reused by other trackers and exhibit high accuracy. The method's consistent results across datasets demonstrate its independence from specific data, clearly distinguishing it from training-based methods. Extending 3D object tracking to non-rigid objects for broader applications and exploring tracking for objects with more complex topologies are future research directions in our work.

Acknowledgments

This work is partially supported by the National Key R&D Program of China under grant (No. 2022YFB3303203), NSF of China (U23A20312, No. 62172260, 62202425), and Shandong Provincial Natural Science Foundation (No. ZR2022LZH007).

References

[1] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. PoseRBPF: A rao-blackwellized particle filter for 6-D object pose tracking. *IEEE Transactions on Robotics*, 37(5):1328–1342, 2021. 5, 6

[2] Xingyi He, Jiaming Sun, Yuang Wang, Di Huang, Hujun Bao, and Xiaowei Zhou. OnePose++: Keypoint-free one-shot object pose estimation without CAD models. In *Advances in Neural Information Processing Systems*, 2022. 2

[3] Hong Huang, Fan Zhong, Yuqing Sun, and Xueying Qin. An occlusion-aware edge-based method for monocular 3D object tracking using edge confidence. *Computer Graphics Forum*, 39(7):399–409, 2020. 2

[4] Hong Huang, Fan Zhong, and Xueying Qin. Pixel-wise weighted region-based 3D object tracking using contour constraints. *TVCG*, 28(12):4319–4331, 2022. 1, 2, 5, 7

[5] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *CVPR*, pages 3907–3916, 2018. 3

[6] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. In *ICCV*, pages 3992–4003, 2023. 2, 4

[7] JongMin Lee, Yohann Cabon, Romain Brégier, Sungjoo Yoo, and Jérôme Revaud. MFOS: Model-free & one-shot object pose estimation. In *AAAI*, pages 2911–2919, 2024. 2

[8] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In *ICCV*, pages 7707–7716, 2019. 2, 3, 4

[9] 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. In *ECCV (32)*, pages 298–315, 2022. 1, 2, 5, 6

[10] Matthew M. Loper and Michael J. Black. OpenDR: An approximate differentiable renderer. In *ECCV*, pages 154–169, 2014. 3

[11] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2022. 3

[12] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136, 2011. 6

[13] Keunhong Park, Arsalan Mousavian, Yu Xiang, and Dieter Fox. Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. *arXiv:1912.00416*, 2019. 5, 6

[14] Sida Peng, Xiaowei Zhou, Yuan Liu, Haotong Lin, Qixing Huang, and Hujun Bao. PVNet: Pixel-wise voting network for 6DoF object pose estimation. *TPAMI*, 44(6):3212–3223, 2022. 5, 6

[15] Denys Rozumnyi, Jirí Matas, Marc Pollefeys, Vittorio Ferrari, and Martin R. Oswald. Tracking by 3D model estimation of unknown objects in videos. In *ICCV*, pages 14040–14050, 2023. 3

[16] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *3DIM*, pages 145–152, 2001. 6

[17] Xiuqiang Song, Weijian Xie, Jiachen Li, Nan Wang, Fan Zhong, Guofeng Zhang, and Xueying Qin. 3D object tracking for rough models. *Computer Graphics Forum*, 42(7), 2023. 2, 7

[18] Manuel Stoiber, Martin Pfanne, Klaus H. Strobl, Rudolph Triebel, and Alin Albu-Schäffer. A sparse gaussian approach to region-based 6DoF object tracking. In *ACCV*, pages 666–682, 2020. 7

[19] Manuel Stoiber, Martin Pfanne, Klaus H. Strobl, Rudolph Triebel, and Alin Albu-Schäffer. SRT3D: A sparse region-based 3D object tracking approach for the real world. *IJCV*, 130(4):1008–1030, 2022. 2, 7

[20] Manuel Stoiber, Martin Sundermeyer, and Rudolph Triebel. Iterative corresponding geometry: Fusing region and depth for highly efficient 3D tracking of textureless objects. In *CVPR*, pages 6855–6865, 2022. 1, 2

[21] Jiaming Sun, Zihao Wang, Siyu Zhang, Xingyi He, Hongcheng Zhao, Guofeng Zhang, and Xiaowei Zhou. OnePose: One-shot object pose estimation without CAD models. In *CVPR*, pages 6825–6834, 2022. 2

[22] Xuhui Tian, Xinran Lin, Fan Zhong, and Xueying Qin. Large-displacement 3D object tracking with hybrid non-local optimization. In *ECCV*, pages 627–643, 2022. 1, 2

[23] Henning Tjaden, Ulrich Schwanecke, Elmar Schömer, and Daniel Cremers. A region-based gauss-newton approach to real-time monocular multiple object tracking. *TPAMI*, 41(8): 1797–1812, 2019. 2, 6, 7

[24] Bin Wang, Fan Zhong, and Xueying Qin. Pose optimization in edge distance field for textureless 3D object tracking. In *CGI*, pages 32:1–32:6, 2017. 2

[25] Bin Wang, Fan Zhong, and Xueying Qin. Robust edge-based 3D object tracking with direction-based pose validation. *Multimedia Tools and Applications*, 78(9):12307–12331, 2019. 2

[26] Chen Wang, Roberto Martín-Martín, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke Zhu. 6-PACK: Category-level 6D pose tracker with anchor-based keypoints. In *ICRA*, pages 10059–10066, 2020. 3

[27] Long Wang, Shen Yan, Jianan Zhen, Yu Liu, Maojun Zhang, Guofeng Zhang, and Xiaowei Zhou. Deep active contours for real-time 6-DoF object tracking. In *ICCV*, pages 13988–13998, 2023. 1, 3

[28] Bowen Wen and Kostas E. Bekris. BundleTrack: 6D pose tracking for novel objects without instance or category-level 3D models. In *IROS*, pages 8067–8074, 2021. 3

[29] Bowen Wen, Jonathan Tremblay, Valts Blukis, Stephen Tyree, Thomas Müller, Alex Evans, Dieter Fox, Jan Kautz, and Stan Birchfield. BundleSDF: Neural 6-DoF tracking and 3D reconstruction of unknown objects. *CVPR*, 2023. 2

[30] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. FoundationPose: Unified 6D pose estimation and tracking of novel objects. In *CVPR*, pages 17868–17879, 2024. 1, 2

[31] Yijia Weng, He Wang, Qiang Zhou, Yuzhe Qin, Yueqi Duan, Qingnan Fan, Baoquan Chen, Hao Su, and Leonidas J. Guibas. CAPTRA: Category-level pose tracking for rigid and articulated objects from point clouds. In *ICCV*, pages 13189–13198, 2021. 3

[32] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *ECCV*, pages 766–782, 2016. 6

[33] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 6