ARKitTrack: A New Diverse Dataset for Tracking Using Mobile RGB-D Data

Haojie Zhao[1†] Junsong Chen[1†] Lijun Wang[1*] Huchuan Lu[1,2]
[1]Dalian University of Technology, China [2]Peng Cheng Laboratory, China
{haojie_zhao,jschen}@mail.dlut.edu.cn {ljwang,lhchuan}@dlut.edu.cn

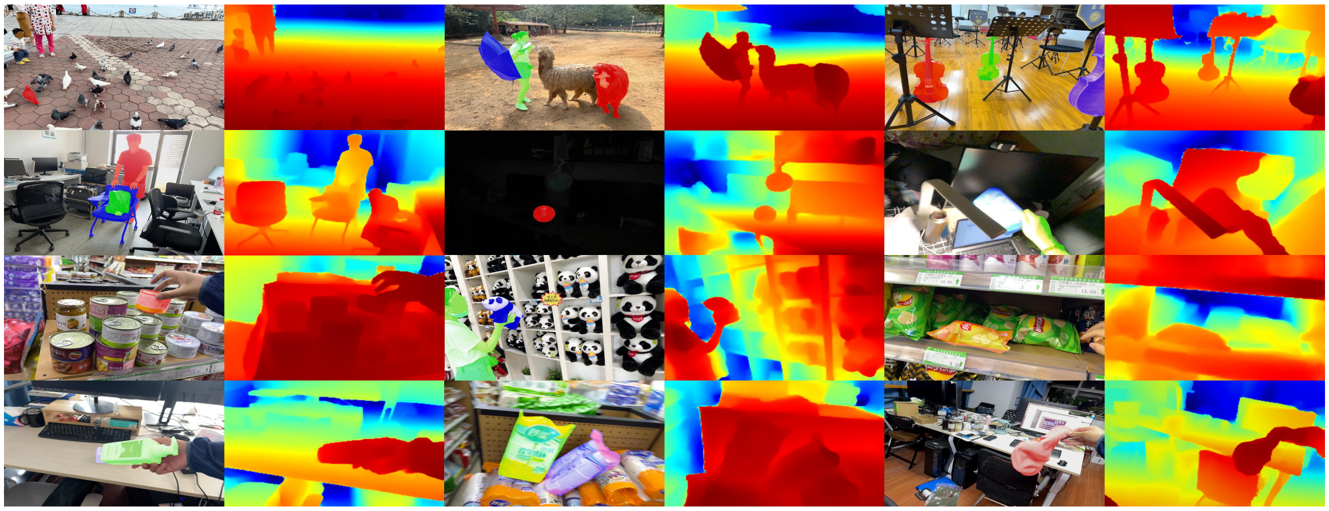
Figure 1. Samples from ARKitTrack. We capture both indoor and outdoor sequences (1st row) in many scenes, including zoo, market, office, square, corridor, etc. Lots of scenarios are presented in our dataset, e.g., low or high light conditions (2nd row), surrounding clutter (3rd row), out-of-plane rotation, motion blur, deformation, etc. (4th row). Besides, we annotate each frame with object masks.

Abstract

Compared with traditional RGB-only visual tracking, few datasets have been constructed for RGB-D tracking. In this paper, we propose ARKitTrack, a new RGB-D tracking dataset for both static and dynamic scenes captured by consumer-grade LiDAR scanners equipped on Apple's iPhone and iPad. ARKitTrack contains 300 RGB-D sequences, 455 targets, and 229.7K video frames in total. Along with the bounding box annotations and frame-level attributes, we also annotate this dataset with 123.9K pixel-level target masks. Besides, the camera intrinsic and camera pose of each frame are provided for future developments. To demonstrate the potential usefulness of this dataset, we further present a unified baseline for both box-level and pixel-level tracking, which integrates RGB features with bird's-eye-view representations to better explore cross-modality 3D geometry. In-depth empirical analysis has verified that the ARKitTrack dataset can significantly facilitate RGB-D tracking and that the proposed baseline method compares favorably against the state of the arts. The code and dataset is available at https://arkittrack.github.io.

†Equal contribution
*Corresponding author: Dr. Lijun Wang, ljwang@dlut.edu.cn

1. Introduction

As a fundamental and longstanding problem in computer vision, visual tracking has been studied for decades and achieved significant progress in recent years with many advanced RGB trackers [6, 7, 9, 20, 37, 39] and datasets [11, 14, 29, 45] being developed. Nonetheless, there still exist many challenging situations such as occlusion, distraction, extreme illumination, etc., which have not been well addressed. With the wide application of commercially available RGB-D sensors, many recent works [15, 33, 42, 54, 55] have focused on the RGB-D tracking problem, as depth can provide additional 3D geometry cues for tracking in complicated environments. The development of RGB-D tracking is always boosted by the emergence of RGB-D tracking datasets. The early RGB-D datasets [36, 43] only have a limited number of video sequences and can hardly meet the requirement of sufficient training and evaluating sophisticated RGB-D trackers. To alleviate this issue, two larger datasets [26, 48] have been built recently and successfully adopted in the VOT-RGBD challenges [16–18].

Though existing RGB-D tracking datasets strongly benefit the development of RGB-D trackers, they are still limited in the following two aspects. First, these datasets are collected using Realsense or Kinect depth cameras, which require edge devices for onsite computing or post-processing and are not easily portable. As a result, it severely restricts the scene diversity, and most videos are captured under static scenes, which causes a large domain gap between the collected dataset and real-world applications. Second, existing RGB-D tracking datasets only contain bounding box-level annotations and mostly fail to provide pixel-level mask labels. Therefore, they are not applicable for training/evaluating pixel-level tracking tasks (i.e., VOS).

With the recent launch of built-in depth sensors of mobile phones (e.g., LiDAR, ToF, and stereo cameras) and the release of AR frameworks (e.g., Apple's ARKit [3] and Google's ARCore [2]), it becomes more convenient than ever to capture depth data under diverse scenes using mobile phones. Compared to prior depth devices, mobile phones are highly portable and more widely used for daily video recording. Besides, the depth maps captured by consumer-grade sensors mounted on mobile phones are also different from previous datasets in terms of resolution, accuracy, etc.

In light of the above observations, we present ARKitTrack, a new RGB-D tracking dataset captured using iPhone built-in LiDAR with the ARKit framework. The dataset contains 300 RGB-D sequences, 229.7K video frames, and 455 targets. Precise box-level target locations, pixel-level target masks, and frame-level attributes are also provided for comprehensive model training and evaluation. Compared to existing RGB-D tracking datasets, ARKitTrack enjoys the following two distinct advantages. First, ARKitTrack covers more diverse scenes captured under both static and dynamic viewpoints. Camera intrinsic and 6-DoF poses estimated using ARKit are also provided for more effective handling of dynamic scenes. Therefore, ARKitTrack is more coincide with real application scenarios, particularly for mobile phones. Second, to our best knowledge, ARKitTrack is one of the first RGB-D tracking datasets annotated with both box-level and pixel-level labels, which is able to benefit both VOT and VOS.

To demonstrate the strong potential of ARKitTrack, we design a general baseline RGB-D tracker, which effectively narrows the gap between visual object tracking (VOT) and segmentation (VOS). Most existing RGB-D trackers employ the low-level appearance cues (e.g., contours and regions) of depth maps but fail to explore the 3D geometry information. To remedy this drawback, we propose to integrate RGB features with bird's-eye-view (BEV) representations through a cross-view feature fusion scheme, where RGB feature is mainly used for target appearance modeling and BEV representations built from depth maps can better capture 3D scene geometry. Experiments on our ARKitTrack datasets demonstrate the merit of the baseline tracker.

Our contribution can be summarized into three folds:

- A new RGB-D tracking dataset, ARKitTrack, containing diverse static and dynamic scenes with both box-level and pixel-level precise annotations.
- A unified baseline method for RGB-D VOT and VOS, combining both RGB and 3D geometry for effective RGB-D tracking.
- In-depth evaluation and analysis of the new dataset and the baseline method, providing new knowledge to promote future study in RGB-D tracking.

2. Related Work

2.1. RGB-D Tracking Datasets

Four datasets have been proposed for the RGB-D general object tracking task. Princeton Tracking Benchmark (PTB) [36] is constructed to alleviate the lack of RGB-D tracking datasets. This dataset is captured with Kinect and contains 100 indoor video sequences. Sequences that have full occlusion frames in this dataset can evaluate the re-detection ability, which is important for long-term tracking. However, this dataset has a calibration problem, approximately 14% of sequences have channel synchronized error and approximately 8% suffer from miss-alignment. Besides, only 5 validation videos have accessible ground truth. The Spatio-Temporal Consistency (STC) dataset [43] uses Asus Xtion to capture the RGB-D videos. To enrich the data diversity, this dataset captures both indoor and outdoor scenarios. Although there are only 36 sequences, this dataset provides binary and statistical attribute annotations for each frame. Color-and-depth general visual object tracking benchmark (CDTB) [26] is proposed for VOT-RGBD challenge [16]. This dataset is constructed for more realistic settings. It has 80 long-term video sequences and the average length is $1,274$, which is approximately six times longer than PTB and STC. In these sequences, the average length of target absence periods is nearly ten times greater than PTB. All these long-term properties make this dataset more challenging and realistic. To increase the data diversity, this dataset captures more dynamic scenarios and more target pose changes. Besides, various devices are used for data collection, including Kinect, ToF-RGB pair, and Stereo Camera. Recent DepthTrack [48] uses Intel Realsense 415

Table 1. Overview of the existing RGB-D VOT datasets and RGB VOS datasets.

Dataset	Num. Tracks	Total Frames	Max. Frames	Avg. Frames	Min. Frames	Num. Mask	Camera Pose	VOS Subset	Train Subset	Num. Attr.	Year
PTB [36]	100	21.5K	-	215	-	-	✗	✗	✗	5	2013
STC [43]	36	9.0K	-	250	-	-	✗	✗	✗	12	2018
CDTB [26]	80	101.9K	2,501	1,274	406	-	✗	✗	✗	13	2019
DepthTrack [48]	200	294.5K	4,069	1,473	143	-	✗	✗	✓	15	2021
DAVIS [31]	193	6.2K	104	69	25	13.5K	✗	✓	✓	0	2017
YoutubeVOS [45]	6459	94.6K	36	27	4	166.2K	✗	✓	✓	0	2019
ARKitTrack	455	229.7K	2,566	765	110	123.9K	✓	✓	✓	16	2022

to capture video sequences. This large-scale dataset contains 150 training sequences and 50 test sequences, with an average sequence length of $1,473$. To further increase data diversity, DepthTrack captures 40 scene types, 90 object types, and annotates each frame with 15 attributes.

2.2. RGB-D Visual Object Tracking

The development of RGB-D trackers is always limited by the related datasets. By constructing the PTB dataset, Song et al. [36] propose two types of RGB-D trackers. The first tracker extracts both the RGB feature and depth feature. To exploit the depth information, this tracker extracts HOG [8] feature on the depth map and 3D features in point cloud space. Then a classifier is trained for detection-based tracking. The second tracker is based on the 3D point cloud and predicts the 3D bounding box. Zhong et al. [55] propose to extract a set of 3D context key points on the dense depth map and use it for collaborative tracking in RGB-D videos. Bibi et al. [4] build the appearance model and motion model in the 3D space. By using the particle filter framework, they propose a part-based sparse tracker. Kart et al. [15] propose a generic RGB-D tracking framework and convert three Discriminative Correlation Filter (DCF) RGB trackers into RGB-D trackers. In this framework, the depth map is used for foreground segmentation, which can provide a strong cue for occlusion. DAL [33] embeds the depth information into deep features and learns a depth-aware discriminative correlation filter. It uses the depth map to generate a modulation map, which can reweight the learned base filter to reduce the effect of the background and occlusion. Besides, the depth histogram is used for the target presence indicator. With this indicator and a re-detection scheme, DAL can perform long-term RGB-D tracking. SiamOC [53] converts SiameseRPN [21] into a RGB-D tracker. This tracker uses the depth map not only for occlusion estimation but also to correct the target location. According to the different distributions of the target's depth histogram, the tracker can estimate the occlusion state. Then, it uses the occlusion state and a Kalman filter to correct the target location. TSDM [54] uses the depth map to generate a target mask to enhance the SiameseRPN++ tracker [20]. Then, depending to the depth value, it cuts the predicted box to give a more precise result. Similarly, Xiao et al. [42] propose a box adaptive strategy according to the depth value. Instead of using the depth map outside the base tracker, recent DeT [48] converts the depth map to a pseudo color image and uses an additional tracker branch to fuse the depth information for tracking.

Most works simply extract the appearance feature from the depth map, and the important geometric cues are ignored. In this work, we explore the geometric information in the BEV space and fuse appearance and geometric features jointly.

2.3. Semi-supervised Video Object Segmentation

Semi-supervised VOS methods aim to transfer the manually labeled frames to the following unlabeled video sequence. A number of matching-based methods [6,12,13,22,25,30,34,35,38,40,41,44] take Semi-VOS as a propagation task. They use the first or intermediate frames as templates to match with future frames. Therefore, how to better use spatial and temporal information in a sequence is always the mainstream research direction. FEELVOS [37] and CFBI [49] perform pixel-level matching globally and locally using both the first frame and previous adjacent frames. STM [30] and the following methods [6, 12, 34, 35, 44] leverage a memory network to store intermediate frames as references and apply non-local attention mechanisms to match and propagate target embeddings. Based on the identification mechanism, AOT [50] can embed multiple target objects into the same embedding space and then propagate all object embeddings uniformly and simultaneously.

All the existing algorithms are developed with only RGB datasets [31, 45]. In this work, we present the first large-scale RGB-D VOS dataset to unveil the potential of RGB-D segmentation. A novel method is also developed with this new dataset, showing satisfactory performance.

3. The ARKitTrack Dataset

3.1. Sequences Collection

Existing RGB-D tracking datasets are captured with depth sensors, such as Kinect and Realsense, which are linked with backend computing devices and not easily portable, resulting in the limited scene diversities. To build a more practical dataset, we collect RGB-D videos using a mobile phone, i.e., iPhone 13 Pro, which has a 12 MP wide-angle color camera and a LiDAR scanner. An iOS APP has also been developed to process and export the captured RGB-D videos at 30 FPS. Each video sequence contains synchronized and aligned RGB frames, depth maps, and confidence maps. The RGB frames are stored using 1920×1440 resolution in JPEG format under a low compression rate. The depth maps and confidence maps are processed with ARKit [3] and stored using 256×192 resolution in 32-bit TIFF and PNG format, respectively. In confidence maps, 2, 1, and 0 indicate high, medium, and low accuracy levels, respectively. To facilitate 3D and AR applications in dynamic scenes, we also provide the camera intrinsic and 6-DoF camera poses of each frame.

During data collection, we ensure scene diversity from the following two aspects. To improve viewpoint diversity, we elaborately capture both static and dynamic scenes. Rich camera motion can cause complicated appearance and depth changes, delivering additional challenges to RGB-D tracking, which mimics real-world application scenarios. To enrich the video content diversity, we capture both indoor videos and outdoor videos under different illumination conditions in a large number of different scenes, including zoo, market, office, street, square, corridor, etc. To further improve tracking difficulties, a large proportion of the sequences contain distracting objects with a similar appearance to the target. Besides the above features, the ARKitTrack dataset also contains many other challenging factors which have been classified into 16 attributes (See Sec. 3.2).

The final ARKitTrack dataset contains 300 sequences with 455 targets, which includes 144 object categories and 287 dynamic scenes, exceeding [48] (90/44) and [26] (21/0). Table 1 compares the overall properties between ARKitTrack and existing RGB-D VOT and VOS datasets. We select 50 test sequences for VOT, which have rich scenes and motion patterns, and are sufficiently challenging for visual tracking. The average length of the VOT test set is $1,286$ frames and therefore can be used for long-term tracking evaluation. For VOS, we also select 55 different test sequences at an average length of 328 frames. These sequences have complicated contents and are longer than many previous VOS sequences, giving more challenges to segmentation algorithms. For each test set, the remaining video sequences are used as the training data (250 and 245 training videos for VOT and VOS, respectively).

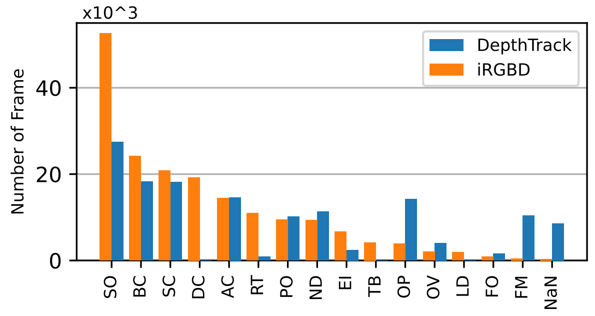

Figure 2. Attribute statistic comparison between ARKitTrack and DepthTrack test set.

3.2. Data Annotation

The proposed dataset has three types of manual annotations, including axis-aligned target bounding boxes, pixel-level target masks, and per-frame attributes. Annotating object masks is always an expensive work. Most previous works (e.g. [45, 52]) annotate masks in a low frame rate or only annotate some keyframes. To annotate object masks in our dataset, we first ask annotators with expertise in VOT and VOS to manually select the appropriate sequences and target objects for annotation, and then uniformly sample key frames for every 3-4 frames, leading to the frame rates of 7-10 FPS. After that, we ask annotators to carefully annotate each keyframe with pixel-level target masks. The bounding box annotations for key frames can be automatically generated based on the target masks (i.e., axis-aligned boxes that most tightly enclose the target regions). Box annotations of remaining frames are generated using the box interpolation method and manually refined by annotators.

We also annotate 16 per-frame challenging attributes for the VOT test sequences to enable attribute-level analysis. Among them, we borrow 12 attributes from prior works [26, 48], including Aspect-ratio Change (AC), Background Clutter (BC), Non-rigid Deformation (ND), Fast Motion (FM), Full Occlusion (FO), Out-of-plane Rotation (OP), Out-of-view (OV), Partial Occlusion (PO), Reflective Targets (RT), Size Change (SC), Similar Objects (SO), and Unassigned (NaN). Another 4 attributes are specifically designed for our dataset, including Depth Clutter (DC), Extreme Illumination (EI), Low Depth Quality (LD), and Target Blur (TB). The LD label is calculated from the confidence map, which is predicted by ARKit and can provide useful information for realistic RGB-D applications. DC, EI, and TB reflect the diversity of scenes in our dataset, where rich motion scenarios cause TB frames. Please refer to the supplementary materials for details. We use the VOT annotation tool [1] to annotate frame-level attributes for the VOT test set. Figure 2 compares the attribute distributions of ARKitTrack and DepthTrack. Our test sequences contain more distracting objects, background clutter, size change, extreme illumination, and reflective target frames.

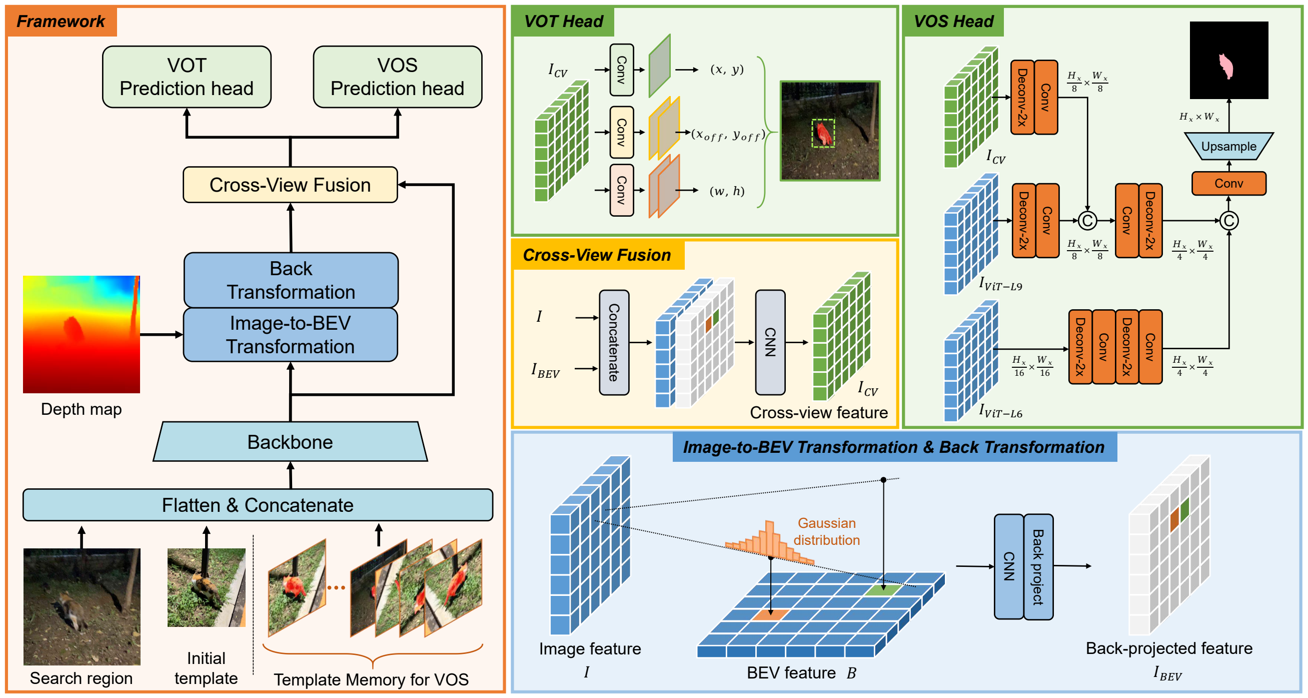

Figure 3. Overview the proposed unified RGB-D tracking pipeline in both box-level (VOT) and pixel-level (VOS).

3.3. Performance metrics

For box-level tracking, we construct a test set (ARKitTrack-VOT) with 50 long-term sequences. Therefore, we use the VOT long-term evaluation protocol [26]. According to this protocol, a tracker is required to perform one-pass tracking on each sequence and predict bounding boxes and confidence scores. The overall evaluation uses three metrics: Precision (Pr.), Recall (Re.), and F-score [26]. Tracking precision uses the intersection-over-union (IoU) between the predicted box and groundtruth to measure the target localization accuracy when the target is visible. Tracking recall can measure the accuracy of visible prediction. Specifically, by setting a confidence threshold τ, all predictions $A(\tau)$ with confidence that higher than the threshold will be used for evaluation, resulting in the following measures

$$Pr(\tau) = \frac{1}{N_p} \sum_{t \in \{t: A_t(\tau) \neq \emptyset\}} IoU(A_t(\tau), G_t), \quad (1)$$

$$Re(\tau) = \frac{1}{N_g} \sum_{t \in \{t: G_t \neq \emptyset\}} IoU(A_t(\tau), G_t), \quad (2)$$

where G is the groundtruth, N_p is the number of non-empty predictions, i.e., $A(\tau) \neq \emptyset$, N_g is the number of visible frames, i.e., $G \neq \emptyset$. The primary metric F-score is defined as

$$F(\tau) = \frac{2Pr(\tau)Re(\tau)}{Pr(\tau) + Re(\tau)}. \quad (3)$$

For each tracker, by computing precision and recall over all possible thresholds, we can draw a precision-recall plot and find the highest F-score as the final measure score.

For pixel-level tracking, we select 55 long-term sequences to construct a VOS test set (ARKitTrack-VOS). On this subset, we use three overall evaluation metrics, including region similarity ($\mathcal{J}$), contour accuracy ($\mathcal{F}$), and their average ($\mathcal{J}\&\mathcal{F} = (\mathcal{J}+\mathcal{F})/2$). The region similarity is defined as the intersection-over-union between the estimated mask and groundtruth mask. The contour accuracy is the F-measure based on the contour-based precision and recall. We calculate the mean value $\mathcal{J}_\mathcal{M}$ and $\mathcal{F}_\mathcal{M}$ over all test sequences as the final scores.

4. A Unified RGB-D Tracking Baseline

We introduce a new RGB-D tracking baseline, which unifies both box-level and pixel-level target tracking. As opposed to existing RGB-D tracking methods that mainly explore the appearance cues in depth maps, we model 3D scene geometry from the BEV view and perform cross-view fusion to integrate appearance and geometry representations for robust RGB-D tracking. As illustrated in Figure 3, the overall pipeline consists of an image-view encoder, BEV transformer, cross-view fusion module, and task-specific inference heads. Architecture designs are detailed as follows.

Table 2. Comparison results on the ARKitTrack test set and existing popular RGB-D benchmarks, including DepthTrack and CDTB.

Tracker	ARKitTrack			DepthTrack			CDTB			Description	
	Pr.	Re.	F-score	Pr.	Re.	F-score	Pr.	Re.	F-score	Type	Year
Start-ST101 [47]	0.407	0.381	0.393	0.503	0.468	0.485	0.657	0.669	0.663	RGB	2022
OSTrack [51]	0.440	0.440	0.440	0.572	0.563	0.567	0.713	0.686	0.699	RGB	2022
MixFormer1k [7]	0.449	0.421	0.434	0.490	0.454	0.471	0.692	0.664	0.678	RGB	2022
ToMP101 [27]	0.449	0.433	0.441	0.515	0.495	0.505	0.670	0.683	0.676	RGB	2022
TSDM [54]	0.389	0.292	0.334	0.442	0.363	0.398	0.647	0.543	0.591	RGBD	2021
ATCAIS [17]	0.389	0.343	0.364	0.473	0.402	0.435	0.709	0.696	0.702	RGBD	2020
DAL [33]	0.446	0.329	0.378	0.512	0.369	0.429	0.620	0.560	0.589	RGBD	2020
TALGD [18]	0.428	0.352	0.386	0.494	0.424	0.456	0.630	0.596	0.613	RGBD	2022
DeT [48]	0.428	0.405	0.416	0.560	0.506	0.532	0.674	0.642	0.657	RGBD	2021
STARK_RGBD [18]	0.469	0.426	0.446	0.570	0.558	0.564	**0.743**	**0.769**	**0.755**	RGBD	2022
DDiMP [17]	**0.495**	0.413	0.450	0.540	0.475	0.506	0.703	0.689	0.696	RGBD	2020
Ours	0.488	**0.469**	**0.478**	**0.617**	**0.607**	**0.612**	0.711	0.671	0.690	RGBD	2022

4.1. Cross-view Fusion

Image-View Encoding. We first use the ViT model [51] to extract image feature map $\mathbf{I}$, which takes the template-search image pair as input and jointly extracts the image feature. Specifically, the initial template $\mathbf{Z} \in \mathbb{R}^{C_z \times H_z \times W_z}$[1] and current frame search region $\mathbf{X} \in \mathbb{R}^{3 \times H_x \times W_x}$ are first split into $P \times P$ patches independently. Then, these patches are flattened and linearly projected into D dimension token embeddings $\mathbf{E}_z \in \mathbb{R}^{N_z \times D}$ and $\mathbf{E}_x \in \mathbb{R}^{N_x \times D}$. After adding position embeddings, all the token embeddings are concatenated and fed into the ViT model [10] for feature extraction by encoding the correlation between the template and search regions. The enhanced search region tokens are taken as the output and reshaped into a 2D feature map $\mathbf{I} \in \mathbb{R}^{C_I \times H_I \times W_I}$.

Image-to-BEV Transformation. The 2D depth maps suffer from a geometric-lossy problem [24], i.e., faraway points in 3D space might be projected into neighboring pixels in the 2D image plane. Encoding the depth map in the 2D form is therefore a sub-optimal way to exploit geometry information. Instead, we transform the depth map into the BEV space and modulate the BEV feature for better depth map encoding. Specifically, we take the encoded RGB feature map $\mathbf{I}$ as input and follow LSS [32] to embed both the RGB and depth information into a BEV feature map $\mathbf{B} \in \mathbb{R}^{C_B \times H_B \times W_B}$ with the pillar format [19] using an accelerated BEV pooling [24] and conv layers. However, different from LSS which predicts the discrete depth distribution, we model depth for each pixel using Gaussian distribution centered at the input depth value.

The BEV space can better capture 3D geometry, where neighboring points are close to each in 3D space with similar depths. Therefore, we directly use a convolutional sub-network to further modulate the BEV feature, which can effectively aggregate information within a 3D local context and also compensates for the imperfect BEV transformation caused by inaccurate depth.

Image-BEV Cross-View Fusion. In order to perform cross-view fusion, the corresponding features under 2D image and BEV views should be first spatially aligned. Since target localization is conducted on the image plane, we back-project the BEV feature to the 2D image space, producing $\mathbf{I}_{BEV} \in \mathbb{R}^{C_B \times H_I \times W_I}$, as illustrated in Figure 3. For a pixel (i, j) on the image plane, we first project it to 3D and compute its BEV space coordinates (k, l, p) according to its depth $d(i, j)$ and camera intrinsic. We then sample the pillar features $B(k, l)$ from the BEV view using nearest neighbor interpolation for efficiency, though more sophisticated sampling techniques are also applicable. The above procedure is conducted for each pixel and the sampled BEV features can be assembled into the well-aligned image-view feature map $\mathbf{I}_{BEV}$.

Finally, The image feature $\mathbf{I}$ and BEV feature $\mathbf{I}_{BEV}$ are fused via concatenation and conv layers to produce the final feature map $\mathbf{I}_{CV}$ which is further used for downstream tracking tasks.

4.2. Bounding Box and Pixel-level Tracking

Our baseline method provides a general framework for both box and pixel-level tracking tasks (i.e., VOT and VOS), where only task-specific heads with minimum architectural modification are required to build upon the fused features.

VOT Head. The VOT head is a fully-convolutional network with three convolution layers, which consumes the cross-view feature $\mathbf{I}_{CV}$ to predict a score map $\mathbf{L} \in$

[1]The initial template is the RGB image ($C_z = 3$) and the concatenation of the RGB image with initial target mask ($C_z = 4$) for VOT and VOS, respectively.

Table 3. Comparison results on the ARKitTrack VOS test set with existing popular VOS methods.

Segmentor	STCN [6]	AOT [50]	RPCM [46]	QDMN [23]	STCN_RGBD	**Ours**
Year	2021	2021	2022	2022	2021	2022
Type	RGB	RGB	RGB	RGB	RGBD	RGBD
$\mathcal{J}\&\mathcal{F}\uparrow$	0.525	0.582	0.509	0.306	0.537	**0.662**
$\mathcal{J}_{\mathcal{M}}\uparrow$	0.489	0.555	0.492	0.276	0.498	**0.625**
$\mathcal{F}_{\mathcal{M}}\uparrow$	0.560	0.627	0.527	0.337	0.575	**0.698**

$\mathbb{R}^{1\times H_I\times W_I}$, an offset map $\mathbf{O}\in\mathbb{R}^{2\times H_I\times W_I}$, and a box size map $\mathbf{S}\in\mathbb{R}^{2\times H_I\times W_I}$. The target center coordinate (x, y) is determined by the peak response at the score map. We then obtain the corresponding center offset (x_{off}, y_{off}) and box size (w, h) from $\mathbf{O}$ and $\mathbf{S}$, respectively. The final target bounding box is located at

$$(x, y, w, h) = (x + x_{off}, y + y_{off}, w, h). \tag{4}$$

VOS Head. To improve pixel-level segmentation accuracy, we follow prior methods [6,30] by using temporal dynamic templates and multi-scale features. Similar to [7,47], we introduce an IoU prediction branch to control the update of dynamic templates. If the predicted IoU is higher than a threshold, we memorize the current frame and mask as a new dynamic template, and the initial template is concatenated with all the dynamic templates for subsequent tracking. As shown in Figure 3, the VOS head built on top of the multi-scale feature map predicts the final target segmentation map $\mathbf{M}$ with convolutional layers interleaved by deconvolution upsampling.

4.3. Training Loss

For the VOT head, we use the weighted focal loss for score map prediction and ℓ_1 loss with generalized IoU loss for offset and box regression:

$$\begin{aligned}\mathcal{L}_{reg} &= \lambda_{giou}\mathcal{L}_{giou} + \lambda_{\ell_1}\mathcal{L}_{\ell_1},\\ \mathcal{L}_{track} &= \mathcal{L}_{focal} + \mathcal{L}_{reg},\end{aligned} \tag{5}$$

where $\lambda_{giou} = 2$, $\lambda_{\ell_1} = 5$ (please refer to [51] for details).

For the VOS head, we use the dice loss [28] and the bootstrapped cross entropy [5] for masks prediction and the ℓ_2 loss for IoU prediction:

$$\begin{aligned}\mathcal{L}_{mask} &= \lambda_{bce}\mathcal{L}_{bce}(\mathbf{M},\hat{\mathbf{M}}) + \lambda_{dice}\mathcal{L}_{dice}(\mathbf{M},\hat{\mathbf{M}}).\\ \mathcal{L}_{seg} &= \mathcal{L}_{mask} + \mathcal{L}_{\ell_2} + \mathcal{L}_{reg}(\mathbf{b},\hat{\mathbf{b}}).\end{aligned} \tag{6}$$

where $\mathbf{M}$ and $\hat{\mathbf{M}}$ are the predicted and groundtruth masks; $\mathbf{b}$ represents the bounding box enclosing the predicted target region and $\hat{\mathbf{b}}$ denotes the groundtruth bounding box. $\lambda_{bce} = 10$ and $\lambda_{dice} = 2$ are the regularization parameters.

5. Experiments

We adopt pre-trained OSTrack384 [51] to initialize the image-view encoder. All the other network parameters are randomly initialized. The VOT and VOS tasks are trained and evaluated separately by following existing experimental protocols [6,7,30,51] without otherwise stated. The trained model runs at 50FPS and 10FPS for VOT and VOS, respectively, at 384 input resolution on a single NVIDIA 3090 GPU. Detailed evaluation results are as follows.

Table 4. The ablation analysis of our VOT method on the ARKitTrack test set and DepthTrack test set.

Components				ARKitTrack			DepthTrack [48]			FLOPs	Params
FT	BEV	CV	GD	Pr.	Re.	F-score	Pr.	Re.	F-score	G	M
				0.441	0.441	0.441	0.572	0.563	0.567	48.4	92.1
✓				0.448	0.438	0.443	0.579	0.569	0.574	48.4	92.1
✓	✓		✓	0.427	0.420	0.423	0.570	0.537	0.553	56.2	104.1
✓	✓	✓		0.479	0.461	0.470	0.612	0.593	0.602	56.7	104.9
✓	✓	✓	✓	**0.488**	**0.469**	**0.479**	**0.617**	**0.607**	**0.612**	56.7	104.9

5.1. Performance Evaluation

Overall Performance of VOT. We compare the proposed method with 7 recent RGB-D trackers and 3 state-of-the-art RGB trackers. The 7 RGB-D tracker includes 3 recent methods (DeT [48], DAL [33], and TSDM [54]) and the top four trackers from the VOT-RGBD 2021 challenge [18] (STARK_RGBD, TALGD, ATCAIS, DDiMP). Among them, ATCAIS and DDiMP are also the top trackers of VOT-RGBD 2020 challenge [17]. The 4 state-of-the-art RGB trackers are ToMP [27], STARK [47], MixFormer [7], and OSTrack [51].

We perform evaluations on our ARKitTrack-VOT test set, DepthTrack test set [48], and CDTB [26], as shown in Table 2. For all trackers, the overall performance on our dataset is always lower than that on DepthTrack and CDTB. It confirms that the proposed ARKitTrack is more challenging than the existing RGB-D tracking datasets. Our tracker performs better than other trackers on both ARKitTrack and DepthTrack, achieving 0.478 and 0.612 F-score, respectively. Although CDTB does not provide a training set and our tracker cannot gain from model learning on this dataset, our method still achieves satisfactory performance. 0.677 F-score can be achieved by training with ARKitTrack, and 0.690 F-score is achieved by training with DepthTrack.

Overall Performance of VOS. Since there is no existing RGB-D VOS method, we select 4 state-of-the-art RGB-VOS methods for comparison on the ARKitTrack-VOS test set, including STCN [6], RPCM [46], AOT (SwinB-L) [50] and QDMN [23]. Besides, We design a variant named STCN_RGBD for RGB-D VOS by adding an additional depth branch to STCN and fusing RGBD features through concatenation. For a fair comparison, all methods are retrained on ARKitTrack-VOS train set without static image pre-training. As shown in Table 3, our method consistently outperforms all the other compared methods in terms of all metrics. Please refer to the supplementary materials for more comparisons.

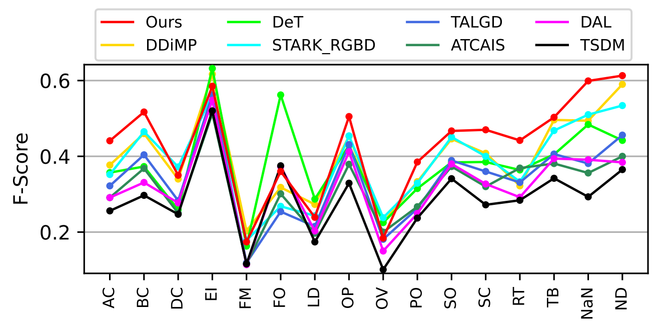

Figure 4. Attribute-specific analysis on the VOT test set.

Attribute-based Performance. We also conduct an attribute-specific analysis for the aforementioned RGB-D VOT trackers by using our per-frame attributes annotation, shown in Figure 4. It shows that our tracker performs better than other RGB-D trackers on 10 attributes. Besides, DeT outperforms other trackers on the full-occlusion attribute. All trackers can deliver satisfactory performance on the extreme-illumination factor. However, none of them can well address the fast-motion and out-of-view attributes, indicating these attributes are more challenging for existing RGB and RGB-D trackers.

5.2. Ablation Study

Studies on VOT. To better demonstrate our method, we conduct ablation studies on both ARKitTrack and DepthTrack, shown in Table 4. Our tracker is trained with their respective training sets and tested on their respective test sets. The basic tracker is OSTrack, which is trained with only the common RGB tracking datasets. By fine-tuning (**FT**) on the corresponding RGB-D training set with only RGB channels, this basic tracker achieves a slight improvement on both test sets. In this work, we propose embedding the depth channel into BEV space (**BEV**) and fusing RGB-D information from cross views (**CV**). By equipping the basic tracker with these methods, the tracker (**FT+BEV+CV**) gains 2.7% and 2.8% F-score on two test sets, respectively. After we apply the Gaussian distribution (**GD**) method in the BEV transformation, the final tracker (**FT+BEV+CV+GD**) further gains about 1% F-score on both test sets. We also attempt to use only the back-projected BEV feature for tracking (**FT+BEV**), but the results are the worst. It shows that the back-projected BEV feature can enhance the image-view feature but is not equivalent to the image-view feature. It also suggests the effectiveness of our cross-view fusion module.

Studies on VOS. We also explore different memory update strategies for VOS, and the results are shown in Table 5. Our method is trained with the ARKitTrack-VOS training set and tested on the ARKitTrack-VOS test set. The baseline is our method that uses the designed VOS head with no dynamic templates achieving 0.630$\mathcal{J}\&\mathcal{F}$. **ONE** indicates that we keep only one template and update it every N frames which improves $\mathcal{J}\&\mathcal{F}$ by 1.2%. By using the IoU prediction branch (**IP**) to control the template update, **IP+ONE** gains 1.8%$\mathcal{J}\&\mathcal{F}$. **ADD** indicates that we consecutively add new templates to the memory bank and maintain several templates for prediction. By using this strategy, **IP+ADD** further gains 1.4%$\mathcal{J}\&\mathcal{F}$ and achieves 0.662$\mathcal{J}\&\mathcal{F}$. These results show the effectiveness of our VOS method and suggest the significance of the temporal information in the VOS task.

Table 5. Comparison with different memorizing strategies.

Components			ARKitTrack		
IP	ONE	ADD	$\mathcal{J}\&\mathcal{F}\uparrow$	$\mathcal{J}_\mathcal{M}\uparrow$	$\mathcal{F}_\mathcal{M}\uparrow$
			0.630	0.594	0.666
	✓		0.642	0.604	0.680
✓	✓		0.648	0.611	0.685
✓		✓	**0.662**	**0.625**	**0.698**

6. Limitation and Ethical Concern

The proposed baseline tracker fails to exploit camera pose information. However, we believe that this information can provide essential cues for target tracking from dynamic viewpoints. Therefore, we will release these pose data along with the ARKitTrack dataset and explore this topic in future studies. We have obtained the approval for all the human targets in our dataset. For background pedestrians, we remove all of the personally identifiable information by blurring their faces for ethical concerns.

7. Conclusion

We present ARKitTrack, a new large-scale RGB-D tracking dataset captured with iPhone built-in LiDAR under diverse scenes. Box annotations, pixel-level target labels, as well as frame-level attributes are provided to facilitate RGB-D VOT and VOS training&evaluation. A general baseline tracker is also designed, which incorporates image appearance with BEV geometry through cross-view fusion and further closes up the gap between RGB-D VOT and VOS. We hope our dataset and baseline method can promote future research for the RGB-D tracking community.

Acknowledgements. This work is supported in part by National Natural Science Foundation of China (62293542, U1903215, 62276045), National Key R&D Program of China (2018AAA0102001), Fundamental Research Funds for the Central Universities (DUT21RC(3)025, DUT22ZD210), and Dalian Science and Technology Talent Innovation Support Plan (2022RY17).

References

[1] Aibu. https://github.com/votchallenge/aibu. 4

[2] Arcore. https://developers.google.com/ar. 2

[3] Arkit. https://developer.apple.com/augmented-reality/. 2, 4

[4] Adel Bibi, Tianzhu Zhang, and Bernard Ghanem. 3d part-based sparse tracker with automatic synchronization and registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1439–1448, 2016. 3

[5] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Modular interactive video object segmentation: Interaction-to-mask, propagation and difference-aware fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 7

[6] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Rethinking space-time networks with improved memory coverage for efficient video object segmentation. In *Advances in Neural Information Processing Systems*, pages 11781–11794, 2021. 1, 3, 7

[7] Yutao Cui, Cheng Jiang, Limin Wang, and Gangshan Wu. Mixformer: End-to-end tracking with iterative mixed attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–12, 2022. 1, 6, 7

[8] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005. 3

[9] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: Efficient convolution operators for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6931–6939, 2017. 1

[10] 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. In *Proceedings of the International Conference on Learning Representations*, pages 1–12, 2021. 6

[11] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. LaSOT: A high-quality benchmark for large-scale single object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5374–5383, 2019. 1

[12] Li Hu, Peng Zhang, Bang Zhang, Pan Pan, Yinghui Xu, and Rong Jin. Learning position and target consistency for memory-based video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4144–4154, 2021. 3

[13] Yuan-Ting Hu, Jia-Bin Huang, and Alexander G. Schwing. Videomatch: Matching based video object segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 56–73, 2018. 3

[14] Lianghua Huang, Xin Zhao, and Kaiqi Huang. GOT-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1562–1577, 2021. 1

[15] Ugur Kart, Joni-Kristian Kämäräinen, and Jiri Matas. How to make an RGBD tracker? In Laura Leal-Taixé and Stefan Roth, editors, *Proceedings of the European Conference on Computer Vision Workshops*, pages 148–161, 2018. 1, 3

[16] Matej Kristan, Amanda Berg, Linyu Zheng, Litu Rout, Luc Van Gool, Luca Bertinetto, Martin Danelljan, et al. The seventh visual object tracking VOT2019 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2206–2241, 2019. 2

[17] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman P. Pflugfelder, Joni-Kristian Kämäräinen, Martin Danelljan, et al. The eighth visual object tracking VOT2020 challenge results. In Adrien Bartoli and Andrea Fusiello, editors, *Proceedings of the European Conference on Computer Vision Workshops*, pages 547–601, 2020. 2, 6, 7

[18] Matej Kristan, Jirí Matas, Ales Leonardis, Michael Felsberg, Roman P. Pflugfelder, Joni-Kristian Kämäräinen, Hyung Jin Chang, Martin Danelljan, et al. The ninth visual object tracking VOT2021 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2711–2738, 2021. 2, 6, 7

[19] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 6

[20] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4282–4291, 2019. 1, 3

[21] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018. 3

[22] Yongqing Liang, Xin Li, Navid H. Jafari, and Jim Chen. Video object segmentation with adaptive feature bank and uncertain-region refinement. In *Advances in Neural Information Processing Systems*, 2020. 3

[23] Yong Liu, Ran Yu, Fei Yin, Xinyuan Zhao, Wei Zhao, Weihao Xia, and Yujiu Yang. Learning quality-aware dynamic memory for video object segmentation. In *Proceedings of the European Conference on Computer Vision*, 2022. 7

[24] Zhijian Liu, Haotian Tang, Alexander Amini, Xingyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. *arXiv*, 2022. 6

[25] Xiankai Lu, Wenguan Wang, Martin Danelljan, Tianfei Zhou, Jianbing Shen, and Luc Van Gool. Video object segmentation with episodic graph memory networks. In *Proceedings of the European Conference on Computer Vision*, pages 661–679, 2020. 3

[26] Alan Lukezic, Ugur Kart, Jani Käpylä, Ahmed Durmush, Joni-Kristian Kamarainen, Jiri Matas, and Matej Kristan. CDTB: A color and depth visual object tracking dataset and benchmark. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10012–10021, 2019. 2, 3, 4, 5, 7

[27] Christoph Mayer, Martin Danelljan, Goutam Bhat, Matthieu Paul, Danda Pani Paudel, Fisher Yu, and Luc Van Gool. Transforming model prediction for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–18, 2022. 6, 7

[28] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *Fourth International Conference on 3D Vision, 3DV 2016, Stanford, CA, USA, October 25-28, 2016*, 2016. 7

[29] Matthias Müller, Adel Bibi, Silvio Giancola, Salman Al-Subaihi, and Bernard Ghanem. TrackingNet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the European Conference on Computer Vision*, pages 310–327, 2018. 1

[30] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9225–9234, 2019. 3, 7

[31] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 3

[32] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Proceedings of the European Conference on Computer Vision*, pages 194–210, 2020. 6

[33] Yanlin Qian, Song Yan, Alan Lukezic, Matej Kristan, Joni-Kristian Kämäräinen, and Jiri Matas. DAL: A deep depth-aware long-term tracker. In *Proceedings of IEEE Conference on Pattern Recognition*, pages 7825–7832, 2020. 1, 3, 6, 7

[34] Hongje Seong, Junhyuk Hyun, and Euntai Kim. Kernelized memory network for video object segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 629–645, 2020. 3

[35] Hongje Seong, Seoung Wug Oh, Joon-Young Lee, Seongwon Lee, Suhyeon Lee, and Euntai Kim. Hierarchical memory matching network for video object segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 12869–12878, 2021. 3

[36] Shuran Song and Jianxiong Xiao. Tracking revisited using RGBD camera: Unified benchmark and baselines. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 233–240, 2013. 2, 3

[37] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. FEELVOS: fast end-to-end embedding learning for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9481–9490, 2019. 1, 3

[38] Haochen Wang, Xiaolong Jiang, Haibing Ren, Yao Hu, and Song Bai. Swiftnet: Real-time video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1296–1305, 2021. 3

[39] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1328–1338, 2019. 1

[40] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1328–1338, 2019. 3

[41] Ziqin Wang, Jun Xu, Li Liu, Fan Zhu, and Ling Shao. Ranet: Ranking attention network for fast video object segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3977–3986, 2019. 3

[42] Feng Xiao, Qiuxia Wu, and Han Huang. Single-scale siamese network based RGB-D object tracking with adaptive bounding boxes. *Neurocomputing*, 451:192–204, 2021. 1, 3

[43] Jingjing Xiao, Rustam Stolkin, Yuqing Gao, and Ales Leonardis. Robust fusion of color and depth data for RGB-D target tracking using adaptive range-invariant depth models and spatio-temporal consistency constraints. *IEEE Transactions on Cybernetics*, 48(8):2485–2499, 2018. 2, 3

[44] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Shengping Zhang, and Wenxiu Sun. Efficient regional memory network for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1286–1295, 2021. 3

[45] Ning Xu, Linjie Yang, Yuchen Fan, Jianchao Yang, Dingcheng Yue, Yuchen Liang, Brian L. Price, Scott Cohen, and Thomas S. Huang. Youtube-vos: Sequence-to-sequence video object segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 603–619, 2018. 1, 3, 4

[46] Xiaohao Xu, Jinglu Wang, Xiao Li, and Yan Lu. Reliable propagation-correction modulation for video object segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. 7

[47] Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10448–10457, 2021. 6, 7

[48] Song Yan, Jinyu Yang, Jani Käpylä, Feng Zheng, Ales Leonardis, and Joni-Kristian Kämäräinen. Depthtrack: Unveiling the power of RGBD tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10705–10713, 2021. 2, 3, 4, 6, 7

[49] Zongxin Yang, Yunchao Wei, and Yi Yang. Collaborative video object segmentation by foreground-background integration. In *Proceedings of the European Conference on Computer Vision*, pages 332–348, 2020. 3

[50] Zongxin Yang, Yunchao Wei, and Yi Yang. Associating objects with transformers for video object segmentation. In *Advances in Neural Information Processing Systems*, pages 2491–2502, 2021. 3, 7

[51] Botao Ye, Hong Chang, Bingpeng Ma, and Shiguang Shan. Joint feature learning and relation modeling for tracking: A one-stream framework. *arXiv*, 2203.11991, 2022. 6, 7

[52] Pengyu Zhang, Jie Zhao, Dong Wang, Huchuan Lu, and Xiang Ruan. Visible-thermal UAV tracking: A large-scale benchmark and new baseline. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8876–8885, 2022. 4

[53] Wenli Zhang, Kun Yang, Yitao Xin, and Rui Meng. An occlusion-aware rgb-d visual object tracking method based on siamese network. In *2020 15th IEEE International Conference on Signal Processing*, volume 1, pages 327–332, 2020. 3

[54] Pengyao Zhao, Quanli Liu, Wei Wang, and Qiang Guo. TSDM: tracking by siamrpn++ with a depth-refiner and a mask-generator. In *Proceedings of IEEE Conference on Pattern Recognition*, pages 670–676, 2020. 1, 3, 6, 7

[55] Bineng Zhong, Yingju Shen, Yan Chen, Weibo Xie, Zhen Cui, Hongbo Zhang, Duansheng Chen, Tian Wang, Xin Liu, Shu-Juan Peng, Jin Gou, Ji-Xiang Du, Jing Wang, and Wenming Zheng. Online learning 3d context for robust visual tracking. *Neurocomputing*, 151:710–718, 2015. 1, 3