A comprehensive review on advances in instance-level 6D object pose tracking

Yanming Wu[a,e,*], Hui Zhang[b], Patrick Vandewalle[c], Peter Slaets[d], Eric Demeester[a,e]

[a]*Automation, Computer Vision & Robotics (ACRO) , KU Leuven, campus Diepenbeek, Belgium*
[b]*Technical Aspects of Multimodal Systems group, Universität Hamburg, Hamburg, Germany*
[c]*Processing Speech and Images (PSI), KU Leuven, Campus De Nayer (Sint-Katelijne-Waver), Belgium*
[d]*Robotics, Automation and Mechatronics (RAM), KU Leuven, campus Groep T Leuven, Belgium*
[e]*Flanders Make @ KU Leuven, B-3001 Heverlee, Belgium*

Abstract

Instance-level 6D object pose tracking involves tracking a known object in 3D space and estimating its six degrees of freedom (6DoF) pose across consecutive images, starting from the initial pose in the first frame. This technology has wide-ranging applications in various fields, including robotics, augmented reality, and human-machine interaction. Over the years, significant progress has been made in this field. Many methods tackle the problem of instance-level 6D pose tracking from RGB images. These techniques can be classified based on their use of keypoints, edges, region information or direct optimization. Additionally, the availability of affordable RGB-D sensors has prompted the utilization of depth data for 6D pose tracking. Another notable advancement is the adoption of deep neural networks, which have shown promising results. Despite these developments, survey studies on the latest advancements in this field are lacking. Therefore, this work aims to fill this gap by providing a comprehensive review of recent progress in instance-level 6D object tracking, covering the aforementioned advancements. This paper provides a detailed examination of metrics, datasets, and methodology employed in this field. Based on the problem modeling approach, methods reviewed in this paper are categorized into optimization-based, learning-based, filtering-based approaches and hybrid approaches that combine various techniques. Furthermore, quantitative results on several publicly available datasets are presented and analyzed, along with applications and open challenges for future research directions.

Keywords: Review, Object tracking, 6D tracking, Deep learning

1. Introduction

6D object pose estimation is a fundamental problem in computer vision and robotics. The goal is to estimate the 3D translation and 3D rotation of an object relative to the camera. 6D object pose estimation finds diverse applications across fields such as robotic manipulation [1, 2, 3], augmented reality [4, 5, 6], and human-machine interaction [7, 8]. In robotic manipulation, an accurate 6D object pose is crucial for planning and executing grasps effectively. Augmented reality applications rely on precise 6D pose estimation to align virtual objects with physical ones in real-world space. In the context of human-machine interaction, 6D object pose estimation plays a pivotal role in facilitating seamless interactions with physical objects. For instance, continuous updates of a hand-held object's position through 6D pose tracking enable its use as a 3D pointer.

Due to its significance, the 6D pose estimation problem has been extensively studied. Most methods estimate the pose of known objects from a single image without exploiting temporal information, an approach often referred to as tracking-by-detection, as opposed to temporal tracking. Early geometry-based methods rely on handcrafted features such as edges, keypoints, or template matching to align a 3D model with an observed image [9, 10]. With the advent of deep learning, learning-based approaches have become dominant. These methods automatically learn robust feature representations from large datasets, achieving strong performance, though at the expense of extensive annotation and high computational demands.

*Corresponding author
Email address: yanming.wu@kuleuven.be (Yanming Wu)

Early deep learning-based methods primarily focus on instance-level pose estimation, which means that they can only be applied to the specific instances on which they are trained [11, 12, 13, 14, 15, 9, 16]. Moreover, they often require the CAD model of the specific instance during inference. Since 2019, category-level approaches [17, 18, 19, 20] have been introduced to address the generalization problem. These methods can estimate poses for unseen objects within the same category without requiring retraining or the exact CAD model of the target instance at inference. However, they still require large amounts of annotated training data for each object class and remain unable to generalize to unseen categories. Recently, several methods for unseen object pose estimation have been proposed [21, 22, 23]. These approaches enhance generalization by enabling pose estimation for unseen objects without the need for retraining, with methods such as FoundationPose [23] demonstrating that transformer-based architectures and large-scale synthetic training can provide strong generalization. Nevertheless, they still require access to the object's CAD model or a few annotated reference images during inference.

A number of notable object pose estimation reviews have been published [24, 25, 26, 27]. Sahin et al. [24] present a comprehensive review of the methods on object pose recovery, from 3D bounding box detectors to full 6D pose estimators. The review for 6D pose estimation is instance-level, covering both classical approaches and deep learning approaches. Prior surveys such as Guan et al. [25], Liu et al. [26], and Wang et al. [27] review only deep learning-based methods for 6D pose estimation. Among them, Guan et al. [25] cover both category-level and instance-level tasks. Liu et al. [26] and Wang et al. [27] further cover unseen object pose estimation.

None of the above surveys addresses the tracking problem explicitly. In contrast, our survey focuses on instance-level 6D object pose tracking, highlighting how temporal information across frames can be exploited for improved performance. In contrast to 6D pose estimation that estimates pose from scratch for each image [11, 12, 13, 14, 15, 28, 29, 9, 16], 6D pose tracking uses the pose estimates from previous frames as a strong prior to aid pose estimation in the current frame. The recursive nature of 6D pose tracking makes pose estimation easier due to a reduced search space, thereby leading to higher computation efficiency. Tracking over time also helps to increase the robustness and coherence of the estimated poses.

Over time, various methods have been proposed for instance-level 6D object tracking from monocular RGB images. These techniques are typically classified based on their use of keypoints, edges, region information or direct optimization. Keypoints-based methods [30, 31] involve detecting and matching keypoints using local features like SIFT [32], and then solving for the object's pose based on these correspondences. Edge-based methods [33, 34, 35] match the projections of the target object's 3D edges to areas of high image gradient, utilizing edges to calculate the relative pose between consecutive frames. Region-based methods [36, 37, 38] utilize pixel-wise color statistics for joint image segmentation and pose estimation, which enables reliable tracking of a wide variety of objects in cluttered scenes while requiring only a texture-less 3D model of the object. Direct optimization methods [39, 40] directly align images between consecutive frames over a 3D target object, assuming brightness (intensity) consistency.

With the availability of affordable RGB-D sensors such as the Microsoft Kinect and Intel Realsense, the interest in the problem of object tracking from depth data has increased greatly. In the last decade, many methods have leveraged depth data for 6D object pose tracking. While purely depth-based pose tracking methods exist, most methods combine information from both depth and RGB images, which generally lead to superior results. Another relatively new development is the use of deep learning techniques, which has demonstrated promising performance. A number of methods adopt an end-to-end architecture to directly regress the relative pose change between two consecutive frames [41, 42, 43]. Other studies incorporate deep neural networks at various stages of the tracking pipeline [44, 45, 46, 47].

Only a few prior surveys address 6D object pose tracking. An earlier work by LePetit et al. [48] extensively examines methods utilizing fiducial markers and those relying on natural features such as edges, texture, or interest points dating back to 2005. A more recent survey conducted in 2019 by Han et al. [49] focuses exclusively on reviewing edge-based methods. Our survey instead aims to present a comprehensive review of non-marker-based, instance-level 6D object pose tracking, covering techniques that exploit monocular RGB, depth, or RGB-D data as input, and including both learning-based and non-learning methods.

The most closely related work is Fan et al. [50], which evaluates deep-learning-based object detection and tracking, with monocular 6D object pose tracking considered only as a minor subtask. In contrast, we provide a dedicated and systematic review of instance-level 6D pose tracking, with detailed analysis of existing methods, datasets, and evaluation metrics. We also offer a broader perspective by reviewing both classical approaches and deep learning-based methods. Classical approaches remain highly relevant: they are lightweight, interpretable, and less dependent on large annotated datasets or high-end hardware, making them attractive for real-time applications and deployment

in resource-constrained environments. In addition, many modern pipelines still integrate classical optimization or filtering components with deep networks to enhance robustness and efficiency, further motivating their inclusion in this survey.

Contributions. Compared to the existing literature, the major contributions of this work can be summarized as follows:

- To the best of our knowledge, this is the first and most recent survey paper to comprehensively cover different aspects for instance-level 6D object tracking, including methods, datasets, evaluation metrics and experimental evaluation among different existing methods.
- This paper provides the reader with the state-of-the-art methods by covering the most recent and advanced progress on instance-level 6D object tracking including deep-learning-based methods.
- Comprehensive performance comparisons of existing methods on several publicly available datasets are provided, with in-depth summaries and insightful discussions being presented.
- Current position of the field on instance-level 6D object tracking is summarized to unveil existing issues. Furthermore, insights on potential future research directions are provided.

Organization. In Section 2, the detailed definition of instance-level 6D object tracking and several most commonly used metrics for evaluating 6D pose tracking algorithms are introduced. Datasets used to evaluate instance-level 6D object tracking methods are presented and compared in Section 3. Methods for instance-level 6D object tracking are reviewed in detail in Section 4 with fine-grained classifications. Section 5 presents the tracking performance analyses on public results. Section 6 presents key applications of instance-level 6D pose tracking. Finally, we discuss future research directions and challenges in Section 7, and conclude our work in Section 8.

Notation. Throughout the paper, scalar variables are represented by lowercase letters (*e.g.*, x), while vector variables are denoted by bold lowercase letters (*e.g.*, $\mathbf{x}$). Capital letters (*e.g.*, P) are adopted for specific functions or variables. Matrix or set of vectors are indicated by bold capital letters (*e.g.*, $\mathbf{I}$). Table 1 lists the symbols utilized in this review. Besides the symbols in the table, there might be additional symbols specific to certain references. As these symbols are not commonly employed, they will be defined within the context in which they appear. Unless otherwise specified, all pose tracking methods in this paper refer to instance-level 6D object pose tracking. To reduce redundancy, we might omit the term "instance-level".

Table 1: Notations employed in this review.

Symbol	Description	Symbol	Description
$\mathbf{I}$	input image	$\mathbf{T}$	transformation matrix
M	3D model	$\Delta\boldsymbol{T}$	relative transform
$\boldsymbol{\mu}$	pose parameter vector	$\Delta\boldsymbol{\mu}$	parameter vector for relative transform
$\Delta\mathbf{t}$	relative translation vector	$\Delta\mathbf{w}$	relative rotation vector
P	probability	E	energy function
k	discrete timestamp	Φ	level-set function

2. Tasks and metrics definition

Before reviewing relevant works, we first define the task of instance-level 6D pose tracking and then introduce several commonly used evaluation metrics in this section.

2.1. Task definition

The objective of instance-level 6D object pose tracking is to compute the 6D pose of an object $\mathbf{T}_k \in \mathbb{SE}(3)$ with respect to the camera coordinate frame at any discrete timestamp $k > 0$, given as input:

- a 3D CAD model or a reconstructed 3D model M of the object instance,
- the initial pose $\mathbf{T}_0 \in \mathbb{SE}(3)$ of the object, typically estimated using a single-image-based 6D pose estimation method,
- a sequence of images $\mathbf{I}_\tau$, $\tau \in \{0, 1, ..., k-1\}$ from previous timestamps and the current observation $\mathbf{I}_k$. Depending on the input modality, $\mathbf{I}_\tau$ can be either RGB image $\mathbf{C}_\tau$, depth image $\mathbf{D}_\tau$, or RGB-D image which is a combination of $\mathbf{C}_\tau$ and $\mathbf{D}_\tau$.

Instance-level 6D object pose tracking can be formulated as a joint probability estimation problem as given below:

$$\boldsymbol{\mu_k}^* = \underset{\boldsymbol{\mu_k}}{\operatorname{argmax}}\, P(\boldsymbol{\mu_k}|\mathbf{I}_0, \mathbf{I}_1, ...\mathbf{I}_{k-1}, \mathbf{I}_k, \mathbf{T}_0, M) \tag{1}$$

Here, $\boldsymbol{\mu_k}$ denotes the parameter vector used to parameterize the transformation $\mathbf{T}_k \in \mathbb{SE}(3)$. The pose $\mathbf{T}$ can be decoupled into rotation $\mathbf{R} \in \mathbb{SO}(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$. While the translation can be straightforwardly represented by the vector $\mathbf{t} = (t_x, t_y, t_z)$, which consists of the x, y, and z components of the object's coordinate frame in the camera coordinate system, the representation of 3D rotation by a 3×3 matrix introduces redundancy and additional constraints. To address this, various compact representations for rotation are commonly used, including Euler angles, quaternions, axis-angle, and Lie algebra representation. Depending on the chosen parameterization, $\boldsymbol{\mu}$ and $\mathbf{T}$ can be readily converted to each other using an appropriate conversion function. Throughout this review, we might use them interchangeably based on the context.

2.2. Metrics definition

To define the mathematical formulation of the evaluation metrics for 6D pose tracking, the following notations are adopted: k denotes the frame index within the evaluation sequence, while N represents the total number of frames. $\mathbf{R}_k$ and $\hat{\mathbf{R}}_k$ denote the ground truth and estimated rotations, respectively, for frame k. Similarly, $\mathbf{t}_k$ and $\hat{\mathbf{t}}_k$ stand for the ground truth and estimated translations for frame k.

2.2.1. Root mean square error

The Root Mean Square (RMS) error calculates the average of each axis component in translation and rotation separately using Equation (2), where $\boldsymbol{\mu}_k = [t_x \quad t_y \quad t_z \quad r_x \quad r_y \quad r_z]^T$ represents the 6D pose parameter for each frame, comprising three translation components and three Euler angles. Here, i denotes one of the six components.

$$e_i(RMS) = \left[\sum_{k=1}^{N} (\boldsymbol{\mu}_k - \hat{\boldsymbol{\mu}}_k)^2 / N\right]^{1/2} \tag{2}$$

2.2.2. Translational and angular error

For each frame k, the translation error is computed as the L_2 norm between the ground truth and estimated translation using Equation (3). The rotation error is defined as an angle derived from Equation (4).

$$e_k(\mathbf{t}) = \| \hat{\mathbf{t}}_k - \mathbf{t}_k \|_2 \tag{3}$$

$$e_k(\mathbf{R}) = \arccos\left(\frac{trace(\mathbf{R}_k \hat{\mathbf{R}}_k^{-1}) - 1}{2}\right) \tag{4}$$

The average translation and angular errors across all frames in the sequence provides a direct method for evaluating tracking performance, as demonstrated in prior research [42].

2.2.3. n cm n° metric

For each frame k, the translation error and angular error are computed as in Equation (3) and (4). A frame is considered successfully tracked if the translation error $e_k(\mathbf{t})$ is below n cm and the angular error $e_k(\mathbf{R})$ is below $n°$. The final accuracy is computed as the tracking success rate across all frames. This evaluation method is commonly referred to as the *n cm n°* metric, where n represents the predefined threshold.

2.2.4. ADD and ADD-S metric

The ADD metric, introduced in [10], quantifies the average distance between 3D model points transformed based on the ground truth pose and those transformed according to the estimated pose. It is a metric dependent on the object model and is formulated as:

$$e_k(ADD) = \frac{1}{m} \sum_{\mathbf{x} \in M} \| (\mathbf{R}\mathbf{x} + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{t}}) \| \tag{5}$$

Here, M denotes the 3D object model and m represents the number of points. For symmetric objects, point matching can be ambiguous in some views. Hence, the ADD metric is often replaced with the ADD-S metric, where the average distance is computed using the closest point distance:

$$e_k(ADD-S) = \frac{1}{m} \sum_{\mathbf{x}_1 \in M} \min_{\mathbf{x}_2 \in M} \| (\mathbf{R}\mathbf{x}_1 + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x}_2 + \hat{\mathbf{t}}) \| \tag{6}$$

The pose for a single frame is considered correctly tracked if $e_k(ADD)$ or $e_k(ADD-S)$ is smaller than e_t, which is a predefined error threshold. The overall tracking quality is assessed using the Area Under Curve (AUC) score. To calculate the AUC, the percentage of correctly tracked poses across all frames is plotted against a range of threshold values from 0 to e_t for each model. The AUC is then obtained by integrating this success rate curve over the specified threshold range. A widely employed value of e_t is $k_e d$, where d is the diameter (i.e., the largest distance between vertices) of M and k_e is a coefficient threshold. A method with higher AUC scores indicates better pose estimation results.

2.2.5. Other metrics

Besides the widely adopted metrics discussed above, several additional 6D pose estimation metrics have been introduced in the BOP challenge [29]. These include Visible Surface Discrepancy (VSD), Maximum Symmetry-Aware Surface Distance (MSSD), and Maximum Symmetry-Aware Projection Distance (MSPD). The overall performance is quantified by the Average Recall (AR), which is computed as the average recall scores across these three error functions. Although these metrics are primarily used for evaluating single-image 6D pose estimation methods, they are in principle also applicable to pose tracking.

3. Datasets

To date, many datasets have been developed for evaluating 6D object pose estimation methods [51, 52, 10, 53, 54, 55, 12]. Building on these efforts, the BOP (Benchmark for 6D Object Pose Estimation) challenge [56] has emerged as the standard evaluation framework. BOP includes multiple widely used datasets, such as LineMOD [10], Occluded LineMOD [51], T-LESS [53], and YCB-Video [12], covering diverse objects under varying levels of clutter and occlusion. It ensures fair comparison by providing datasets in a unified format, standardized evaluation protocols, and consistent error metrics. However, most datasets included in BOP are single-frame based and lack consecutive sequences with frame-by-frame annotations, which makes them unsuitable for benchmarking 6D pose tracking methods.

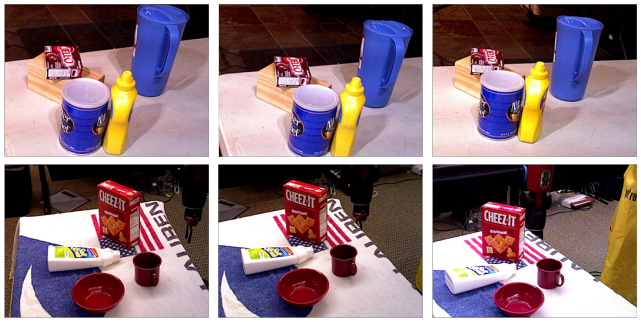
Early instance-level 6D pose tracking algorithms typically employed custom-collected video sequences for testing [57, 58, 59, 36, 60]. These datasets often have limited scale, making it difficult to accurately evaluate the performance of the algorithm. In recent years, some large-scale 6D object pose tracking datasets [12, 42, 61, 37, 62] have been released, enabling more systematic evaluation. In this section, we first introduce the commonly used mainstream instance-level 6D object pose tracking datasets, then provide visual examples of each dataset in Figure 1, and compare them on several important aspects.

3.1. Choi

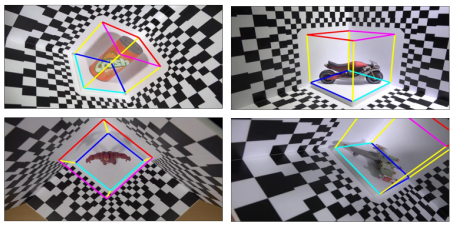
The dataset introduced by Choi and Christensen in 2013 [59] comprises four textured 3D models and four synthetic RGB-D test sequences, each containing 1000 frames. The test sequences are generated by placing each of the four objects in a static texture-less 3D scene and slowly moving the virtual camera around them. The camera trajectories exhibit variations in translation, rotation, and velocity of the camera. The provided ground truth camera trajectory is



(a) Samples in the Choi dataset.



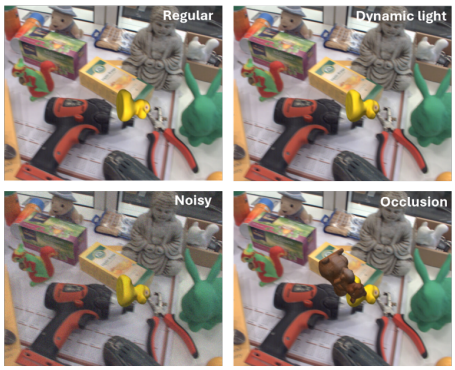
(b) Sample RGB images from the sequences in the YCB-Video dataset.



(c) Samples in the OPT dataset with 6DoF ground-truth pose annotation



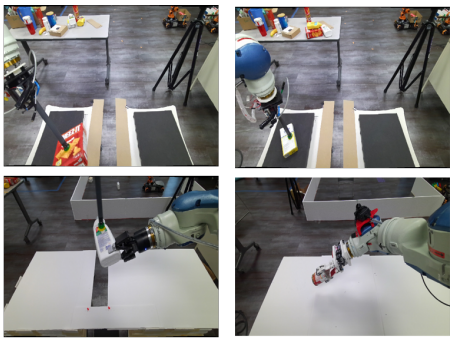
(d) Samples in the Laval 6DOF object tracking dataset.



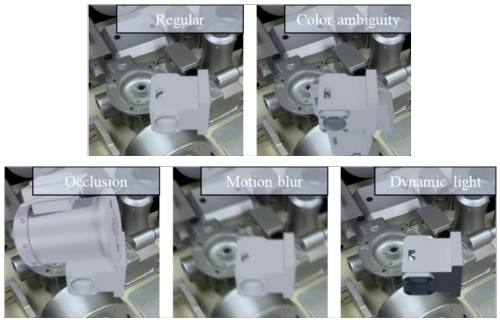
(e) An example frame from the RBOT dataset with the duck model in the four different variants



(f) Samples in the BCOT benchmark, where the blue contour is rendered according to the annotated pose



(g) Samples in the YCBInEOAT dataset



(h) Samples in the InOT6D dataset

Figure 1: Image examples of commonly adopted datasets for instance-level 6D pose tracking.

Table 2: A comparison of datasets for instance-level 6D object pose tracking.

Dataset	Data type	Modality	Motion	Annotation	Texture	Scene attributes	#Objects	#Sequences	#Frames
Choi [59]	synthetic	RGB-D	camera	pre-defined	wl	SO	4	4	4k
YCB-Video [12]	real	RGB-D	camera	depth alignment	wl,wk	MO&OC&DL	21	92	134k
OPT [61]	real	RGB-D	camera	checkerboard	wl	SO&MB&DL	6	138	18k
Laval 6DOF [42]	real	RGB-D	object	marker on objects	wl,wk,tl	SO&OC	11	297	62k
RBOT [37]	semi-synthetic	RGB	camera,object	pre-defined	wl,wk	SO&MO&OC&BC&DL	18	72	72k
BCOT [62]	real	RGB	camera,object	multi-view optimization	tl	SO&MO&OC&BC&MB&DL	20	404	126k
YCBInEOAT [43]	real	RGB-D	object	manual annotation	wl	SO&OC	5	9	7.4k
InOT6D [63]	semi-synthetic	RGB	camera, object	pre-defined	tl	SO&OC&BC&MB&DL&CS	18	90	90k

texture: wl–well-textured wk–weakly-textured tl–texture-less
scene attributes: SO–single object MO–multiple objects OC–occlusion BC–background clutter MB–motion blur DL–dynamic light CS–color similarity

error-free, as the scene is generated through rendering and the camera trajectory is preset. Here, except the respective object itself, the surrounding scene is very simple and texture-less, resulting in highly unrealistic RGB color images overall. This synthetic dataset also lacks distortions in both RGB-D images and motion blurs. Nonetheless, many previous studies [59, 57, 64, 58, 65, 66, 67] on instance-level 6D pose tracking evaluated their approaches using this dataset for quantitative comparison due to the absence of a real 6D pose tracking dataset at that time.

3.2. YCB-Video

The YCB-Video dataset [12] comprises 6D poses of 21 objects from the YCB object set [68] across 92 RGB-D videos, totaling 133,827 frames. Each video features three to nine YCB objects positioned statically on a planar surface and captured by a moving Asus Xtion Pro Live RGB-D camera under varying lighting conditions, with each image having a resolution of 640×480 pixels. Objects in this dataset exhibit different symmetries and are arranged in various poses and spatial configurations, resulting in substantial occlusions. Detecting object pose in this dataset is challenging due to varying lighting conditions, significant image noise, and severe occlusions. To obtain the ground truth pose, manual pose annotation is performed for each object in the first frame of each video, and subsequent poses are computed based on the depth map. Some previous works [67, 62] have noted that the pose annotation of YCB-Video [12] is rather inaccurate due to significant depth errors and color-depth misalignments, making it less reliable for high-precision tracking. The official YCB-Video dataset [12] divides the 92 videos into 80 sequences for training and 12 for testing, and the inclusion of additional training sequences has led to widespread utilization of the dataset in deep learning-based methods.

3.3. OPT

The OPT dataset [61] includes six 3D objects that vary in texture and complexity. For each object, 23 scenes were recorded with a Kinect v2 sensor mounted on a robot arm, which moves around the object at different speeds and under varying lighting conditions. Based on the camera moving patterns and lighting conditions, the sequences are classified into seven categories: translation, forward and backward, in-plane rotation, out-of-plane rotation, flashing light, moving light, and free motion. The video sequences are provided in two resolution formats, 1920×1080 and 512×424. In all sequences the object is placed statically in front of an entirely white background surrounded by checkerboards with a black-and-white marker pattern. The ground truth object pose is obtained by tracking the checkerboard points as a reference pose. Since the relative pose between markers and the object remains static at all times, deriving the object pose is straightforward.

3.4. Laval 6DOF

The Laval 6DOF Object Tracking Dataset [42] comprises 297 RGB-D (Kinect V2) sequences with ground truth 6D pose annotation for 11 objects, 27 videos for each object. These sequences are categorized into three different scenarios: stability, occlusion, and interaction. In the stability scenario, the object is kept static and placed at various angles and distances from the camera, aiming at quantifying the degree of jitter in a tracker. In the occlusion scenario, the object rotates on a turntable and progressively becomes occluded by a flat panel. The occlusion level ranges from 0% (unoccluded) to 75%, thereby resulting in very challenging situations for testing trackers. The interaction scenario

involves a person freely moving the object in front of the camera, creating occlusions and variations in object speed. High-resolution 3D models with geometry and texture are also provided. Ground truth object poses are captured using a ViconTM MX-T40 motion capture system. Small retroreflective Vicon markers are tracked during dataset collection and removed automatically in a post-processing step, leading to marker-less final images.

3.5. RBOT

The RBOT dataset [37] comprises 18 different objects, all available as textured 3D triangle meshes. For each model, four variants of a semi-synthetic image sequence with increasing complexity are generated, namely regular, dynamic light, noisy and occlusion. Each of the 72 video sequences contains 1001 RGB frames in size 640×512. For all sequences, the textured 3D model is rendered using a predefined trajectory of continous 6DoF motion and composited onto a real image sequence providing for a background in each frame. The regular variant simulates a moving object in front of a static camera using a static point light source. The dynamic light variant simulates simultaneous motion of both the object and the camera utilizing a dynamic light source. The noisy variant further distorts the rendering by adding artificial Gaussian noise. The occlusion variant contains an additional second object occluding the first object under dynamic light source.

3.6. BCOT

The BCOT (BinoCular Object Tracking) benchmark [62] features accurately annotated real videos, where both the camera and objects can move freely. This dataset contains 20 objects, 22 scenes, 404 RGB video sequences and 126K images, captured from two orthogonal views using the high-resolution, high-speed and synchronous cameras MER-131-210U3. The set of 20 texture-less objects includes irregular objects, hollow objects and symmetrical objects. The 22 scenes are composed of a variety of scene attributes and multiple motion modes: static camera set, movable camera set, indoor scenes, outdoor scenes, movements in the form of translation, suspension and handheld, dynamic light and occlusion. The object pose is solved by jointly optimizing the shape re-projection error of multiple views in an object-centered pose estimation framework. The annotation error is guaranteed to be less than 2 mm, achieving the best at present. The image resolution is 640×512.

3.7. YCBInEOAT

Wen et al. [43] introduced the YCBInEOAT dataset in realistic, dynamic robotic manipulation scenarios. Unlike the YCB-Video dataset, where objects remain static on a tabletop, YCBInEOAT captures dynamic object motions during real robot operations. It includes 9 RGB-D video sequences collected with an Azure Kinect sensor (20–30 Hz) mounted statically. At the same time, a dual-arm Yaskawa Motoman SDA10f manipulates 5 widely used YCB objects: bleach cleanser, cracker box, mustard bottle, sugar box, tomato soup can. Three types of manipulation tasks are recorded: (1) single-arm pick-and-place, (2) within-hand manipulation, and (3) hand-off between arms. Various robot end-effectors are used, including a vacuum gripper, a Robotiq 2F-85 gripper, and a Yale T42 hand. Ground-truth 6D object poses are manually annotated for each frame.

3.8. InOT6D

The InOT6D dataset [63] was developed to benchmark 6D object pose tracking in complex industrial environments. It contains 18 objects collected from T-LESS [53], MVTec ITODD [69], and additional industrial scenarios. 3D models of these objects are rendered following trajectories similar to the RBOT dataset [37], and the rendered images are composited onto real backgrounds from dynamic industrial scenes. To capture diverse conditions, five variant sequences are generated for each object: regular (REG), dynamic lighting (DYN), occlusion (OCC), motion blur (MOT), and color ambiguity (COR). REG consists of images rendered with a static point light source, while DYN introduces dynamic lighting. OCC adds an orbiting distractor object to simulate occlusion, MOT applies motion blur to the REG sequence, and COR introduces visually similar objects as background distractors. Each variant contains 1001 RGB frames at 640×512 resolution, yielding a total of 90,090 frames across the dataset. For the OCC variant, occlusions are further categorized into modelled (MOC) and unmodelled (UOC) depending on whether the occluding object is included in the dataset.

3.9. Dataset comparison

Table 2 provides an overview and comprehensive comparison of the reviewed datasets, while Figure 1 shows visualization examples. When choosing a benchmark for 6D pose tracking, it is essential to take into account various crucial characteristics, as elaborated upon in the following discussion.

One important feature is the motion type included in the respective image sequences, i.e., whether they involve movement of only the camera, only the object or both simultaneously. In sequences of Choi [59] , YCB-Video [12] and OPT [61], only the camera is moving and they lack extreme object rotations such as out-of-image-plane flipping. The static environment also means that the whole image can be used to solve the camera pose. Consequently, evaluating on these datasets may not fully capture the attributes of a 6D pose tracker.

Another essential aspect is how ground truth annotation is obtained for each frame. Annotating 6D pose for a large number of frames with high precision is challenging. Choi [59], RBOT [37] and InOT6D [63] provide perfect annotation because these are (semi-)synthetic datasets and the object/camera pose is predefined. However, synthetic datasets are not realistic due to the lack of real image distortions and sensor noise. On the other hand, real captured datasets typically use fiducial markers placed in the scene to provide a reference pose. In OPT [61], checkerboards with black and white patterns are placed around a static object, covering a large portion of the captured image with markers. In the Laval 6D object pose tracking dataset [42], small markers are attached to the objects, tracked during object movement, and then removed in the final images after pose annotation. Nonetheless, the visual cues introduced by the marker might still bias the tracker. The YCB-Video [12] and BCOT [37] are both real datasets without markers. YCB-Video [12] uses depth alignment to track the 6D object pose, leading to a significant pose error that prevents its use in high-precision scenarios, such as AR applications. BCOT [62] provides highly accurate pose annotation by leveraging multi-view optimization in an object-centered pose estimation framework, achieving a maximum of 2 mm annotation error.

An additional aspect is whether the dataset simulates common problems encountered in real scenarios, such as diverse object textures, single object (**SO**) or multiple objects (**MO**), occlusions (**OC**), varying degrees of background clutter (**BC**), motion blur (**MB**), dynamic light (**DL**) sources and color similarity (**CS**). Object texture can be classified into 3 categories: well-textured (**wl**), weakly-textured (**wk**) and texture-less (**tl**). Another crucial factor when choosing a 6D pose tracking benchmark is the scale of a dataset. Larger datasets that comprise a wider range of scenarios, variations, and frames provide a more comprehensive and representative assessment of a tracker's capabilities. The object texture, scene attributes and scales for each dataset are shown in Table 2.

4. Approaches for instance-level 6D object tracking

In this paper, we classify the methods under review based on their mathematical modeling approaches. Methods that optimize pose parameters by minimizing an energy function fall into the category of optimization-based approaches. The learning-based approach trains a regressor to learn the parameters characterizing the object's motion from the corresponding data. The filtering-based approach formulates 6D pose tracking as a state estimation problem in a Bayesian filtering framework. Additionally, there exists a hybrid approach that combines different approaches to enhance tracking accuracy and robustness. Together, these four categories—optimization-based, learning-based, filtering-based, and hybrid—provide a discriminative taxonomy that supports a systematic review of advancements in instance-level 6D object pose tracking. In the following sections, we examine each category in detail. For each category, we first outline its working principle, then subdivide methods based on specific characteristics, and finally review the methods within each sub-category. The complete taxonomy of instance-level 6D pose tracking approaches is illustrated in Fig. 2.

4.1. Optimization-based approach

Figure 3 presents a schematic overview of the optimization-based approach. The inputs include the pose variable computed at the previous timestamp $\boldsymbol{\mu}_{k-1}$, the object model M and the current observation $\mathbf{I}_k$, while the output is the pose variable at the current timestamp $\boldsymbol{\mu}_k$.

In the initialization phase, the optimization variable is set to the previous pose variable $\boldsymbol{\mu}_{k-1}$. Then, an energy function is defined to quantify the alignment between the object model and the observed image. This function measures the discrepancy between the model's appearance (as rendered or projected from the estimated pose $\boldsymbol{\mu}$) and the

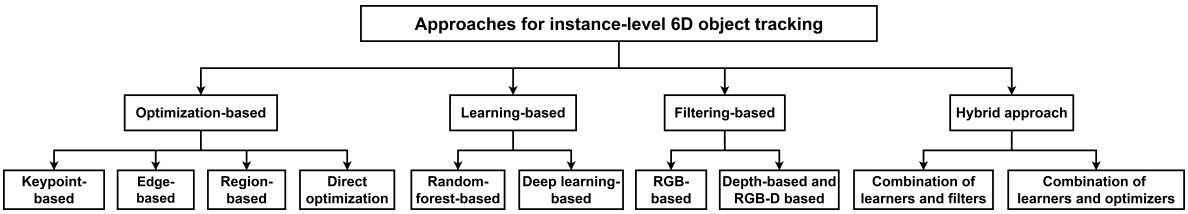


Figure 2: A taxonomy of instance-level 6D object tracking approaches. Methods are grouped into four main categories: optimization-based, learning-based, filtering-based, and hybrid approaches. Optimization-based methods include keypoint-based, edge-based, region-based, and direct optimization approaches. Learning-based methods are divided into random-forest-based and deep-learning-based approaches. Filtering-based methods are split into RGB-based and depth-/RGB-D-based approaches. Hybrid methods combine learners with optimizers or filters.

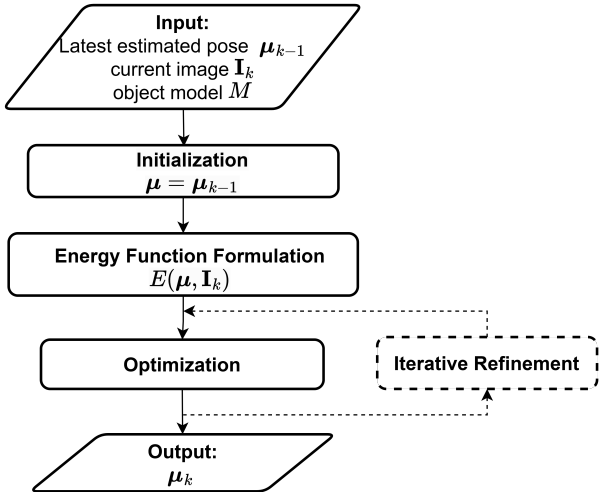


Figure 3: Overall schematic representation of optimization-based approach.

actual appearance of the object in the current frame $\mathbf{I}_k$. Finally, the pose variables are solved by minimizing the energy function. Depending on the formulation of the energy function, it is sometimes possible to obtain an analytical solution for the pose variables. A more general solution involves adjusting the pose variables iteratively to reduce the energy function over iterations. This can be achieved using gradient-based optimization techniques such as gradient descent. This iterative optimization process continues until the convergence criterion is met, such as reaching a predefined threshold for the energy function or a maximum number of iterations. This schematic representation describes the optimization steps for each individual timestamp. To achieve pose tracking across all frames, this optimization procedure is repeated for each timestamp, utilizing the current estimated pose variables to initialize pose variables for the next timestamp.

All optimization-based methods follow the diagram outlined above, differing primarily in how the energy function is formulated and in the optimization techniques employed. Based on the image information utilized for constructing the energy function, existing optimization-based methods can be classified into four major categories: keypoint-based, edge-based, region-based and direct optimization.

4.1.1. Keypoint-based

Keypoint-based methods rely on establishing correspondences between individual features across images for tracking purposes. They have been widely employed for model-based 3D object tracking due to their robustness against fast motion, partial occlusions, and illumination variance. However, the reliance on rich texture limits the applicability of keypoint-based methods. They struggle to track texture-less objects because keypoint detection and matching require texture.

Keypoint-based methods typically involve two stages: an offline stage and an online stage. In the offline stage, a set of reference images (keyframes) with known poses of the target object is generated, and a database of object

feature points is constructed by extracting keypoints from these reference images. During the online tracking stage, local features are extracted from the current frame and matched against features from the keyframe with a similar viewpoint, resulting in a set of 2D/3D correspondences.

Commonly used keypoint detectors include SIFT [32], ORB [70], and BRISK [71]. Matching keypoints involves comparing their descriptors using distance metrics such as Euclidean distance or cosine similarity to assess the similarity between descriptors. Keypoints whose descriptors exhibit maximal similarity are identified as corresponding points. In addition to local feature matching based on feature descriptor distance for matching points across images [72, 30, 73, 74], an alternative approach is to use the Kanade-Lucas-Tomasi (KLT) tracker [75, 76, 77]. This method extracts keypoints from the initial image and then tracks them in subsequent images based on optical flow [78, 31, 79]. While KLT handles continuity better and keeps tracking points that cannot be detected as keypoints, it cannot handle the appearance and disappearance of keypoints.

Once the correspondence between keypoints is established, the optimal object pose is solved by minimizing the distances between $\mathbf{x}_i^{\boldsymbol{\mu}}$, which represents the projection of 3D object features at object pose $\boldsymbol{\mu}$, and the corresponding keypoints $\mathbf{s}_i$ observed in the current frame:

$$\boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu}} \sum_{i=1}^{N} \| \mathbf{x}_i^{\boldsymbol{\mu}} - \mathbf{s}_i \|^2 \tag{7}$$

Solving the pose from N pairs of correspondences is referred to as the PnP (Perspective-n-Point) problem. PnP can be solved by either non-iterative approaches that do not require an initial pose such as Direct Linear Transform (DLT) [80], EPnP [81] or iterative approaches that need an initial pose and iteratively optimize the pose parameters based on gradients. The latter is considered as the "gold standard" solution [82]. To mitigate the influence of outliers, such as false matches, robust estimators or RANSAC [83] are commonly employed.

By utilizing reference images with known poses, keypoint-based methods can achieve wide-baseline matching, eliminating the need for an initial pose for initialization. However, the estimated pose might be jerky due to the lack of temporal consistency. Wide-baseline matching may also suffer from reduced accuracy due to fewer correspondences established when there is a significant difference in viewpoints between the reference image and the current image. In contrast, frame-to-frame tracking approaches match keypoints across consecutive frames to compute the relative pose between them. The current pose is then determined by combining transformations between adjacent frames in the sequence. These methods have a higher chance of establishing more correspondences due to the closer viewpoints. Nevertheless, the incremental nature of these methods means that pose errors accumulate over time.

To address the aforementioned challenges, many keypoint-based 3D tracking approaches leverage both keyframes and temporal information from previous frames [84, 30, 85, 74, 79]. In [84], the camera pose is estimated by matching the current frame with two stored reference frames. The camera pose relative to these reference frames is computed by minimizing a cost function based on two-view epipolar geometry and three-view constraints on keypoint locations. To maintain temporal consistency in the estimated poses, previously computed poses within the time-series are utilized to initialize the minimization process and to regularize the error surface, especially in cases where data from the current frame is limited. [30] combines the strengths of frame-to-frame tracking and matching against keyframes to eliminate drift and jitter. To estimate the object's pose, feature matching is performed between the reference image and the current image, as well as between the previous image and the current image. While matching between the current image and the reference image, an intermediate image is synthesized from the keyframe by warping the stored image patches to the last known viewpoint. This process brings the intermediate and current images into close alignment, facilitating easier and more accurate keypoint matching. Several methods [85, 74, 79] combine detection and frame-to-frame tracking to achieve real-time tracking for multiple 3D objects.

4.1.2. Edge-based

Edges are commonly found in industrial settings, indoors, and in urban environments. They provide a high level of invariance to illumination changes and can be detected easily even in the presence of noise or blur, making them highly suitable for visual tracking.

Similar to keypoint-based methods, edge-based methods use the correspondences between 3D object model points and 2D image points as constraints to solve the pose. Given a 3D model and an initial object pose, a set of N 3D model points $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_N\}$ can be obtained by sampling pixels on the projected object contour and backprojecting them

Table 3: Overview of edge-based instance-level 6D object pose tracking methods.

Method	Year	Input	Category	Information	Real-time capable?
RAPID [33]	1990	RGB	First 3D edge tracking	Edge	✓
Marchand [86]	2001	RGB	Edge orientation	Edge	✓
Drummond [34]	2002	RGB	Multiple edge hypotheses	Edge	✓
Vacchetti [87]	2004	RGB	Multiple edge hypotheses	Edge+Keypoint	✓
Wuest [88]	2005	RGB	Multiple edge hypotheses	Edge	✓
Seo [89]	2013	RGB	Region color	Edge+Region	✓
Wang [60]	2015	RGB	Region color	Edge+Region	✓
Imperoli [90]	2015	RGB	Edge orientation	Edge	✓
Bugaev [91]	2018	RGB	Advanced optimization	Edge+Keypoint	✗
Wang [92]	2019	RGB	Edge orientation	Edge	✗
Huang [93]	2020	RGB	Region color	Edge+Region	✓
Sun [94]	2021	RGB	Region color	Edge+Region	✓
Li [95]	2021	RGB	Region color	Edge+Region	✓
Li [96]	2021	RGB	Region color	Edge+Region	✓
LDT3D [97]	2022	RGB	Advanced optimization	Edge+Region	✓

to the surface of the 3D model. The projection of $\mathbf{q}_i$ onto the image plane at object pose $\boldsymbol{\mu}$ is denoted by $\mathbf{x}_i^{\boldsymbol{\mu}}$. The optimal object pose is then solved by minimizing the distance between the projected contour points $\mathbf{x}_i^{\boldsymbol{\mu}}$ and the image contour points $\mathbf{s}_i$, as described in Equation (7).

The key to this minimization scheme lies in accurately matching the 2D edge points and 3D contour points. Unlike keypoints, which have highly distinctive features for robust matching, edges often appear very similar. This similarity can cause ambiguities when different edges are close to each other, potentially leading to false correspondence matching and tracking failures.

The RAPID system [33] was the first edge-based method for 3D tracking. It establishes a 1D search line perpendicular to the projected contour point and finds the image edge point with maximum gradient as the corresponding point. However, in complex backgrounds, using the maximum gradient response as the only criterium to determine the corresponding image edge point can result in many incorrect matches. Since the development of RAPID [33], many improvements have been proposed to enhance the accuracy and robustness of edge-based 3D object tracking. These methods can be classified into five primary categories based on the utilization of edge orientation, multiple edge hypotheses, region color, advanced optimization and multiple pose hypotheses. Table 3 provides an overview of representative edge-based methods, which we explain in more detail below.

4.1.2.1. Edge orientation. RAPID [33] exploits only edge intensity to locate the corresponding image edge point of a 3D model point. To locate the image edge point more accurately, [86] utilizes both edge intensity and edge orientation. A precomputed convolution kernel function of the contour orientation is used to find the image edge point with an orientation similar to the projected contour orientation, rather than the edge point with the maximum gradient in the scanline.

Instead of searching for explicit 3D-2D correspondences, Imperoli et al. [90] directly minimize the holistic distance between the projected object contour and the query image. Pose is optimized by fitting the set of 3D contour points to the edge distance field $\mathbf{D} : \mathbb{R}^2 \rightarrow \mathbb{R}$, computed from a 3D distance transform known as Directional Chamfer Distance (DCD). For each image pixel $\mathbf{x}$, $\mathbf{D}(\mathbf{x})$ represents the minimum distance to an edge point in a joint location and orientation space. For a 3D contour point $\mathbf{q}_i$, the matching cost $\mathbf{e}_i$ is calculated as $\mathbf{e}_i = \mathbf{D}(\mathbf{x}_i^{\boldsymbol{\mu}})$, where $\mathbf{x}_i^{\boldsymbol{\mu}}$ is the 2D projection of $\mathbf{q}_i$. The final energy function is the squared sum of all contour points' matching costs, given by $E(\boldsymbol{\mu}) = \sum_{i=1}^{N} \mathbf{e}_i^2$.

Successive work [92] also exploits both edge distance and direction for pose tracking. The distance-based pose optimization provides a fast and accurate initial estimation, which is then validated with edge direction for increased robustness. Point-wise edge direction consistency is incorporated into the energy function to adaptively weight the matches for robust estimation. To deal with fast motion, a particle filter approach incorporating distance-based estimation and direction-based validation is proposed. Furthermore, direction-based validation is employed to detect tracking failure, and a pose recovery strategy is introduced. The separation of estimation and validation processes

makes the direction-based validation results more reliable compared to those in D^2CO [90], where direction-based validation might be biased due to the joint minimization of distance and direction costs.

4.1.2.2. Multiple edge hypotheses. To address ambiguities in edge matching within complex backgrounds, several researchers have explored the use of multiple edge hypotheses [34, 87, 88]. Drummond and Cipolla [34] incorporate these hypotheses into a robust M-estimator. They identify multiple candidate matching points for each projected contour point, based on intensity discontinuities that exceed a certain threshold. To minimize matching errors, the optimization process assigns lower weights to matches that have more candidate points along the search line. Although multiple candidate edge points are identified in [34], for each sampled 3D edge point, only one edge point is involved in the optimization. In [87] and [88], different hypotheses corresponding to local extrema of the image gradient along the search line are memorized. The robust estimator considers all the potential matches, but only the hypothesis with the lowest residual overall influences the final energy function optimization. Additionally, Vacchetti et al. [87] enhance their approach by combining edge information and keypoints, thus increasing robustness for both textured and non-textured objects. While using multiple edge hypotheses improves tracking robustness, it does so at the expense of increased computational time.

4.1.2.3. Region color. The aforementioned methods search image contour points mainly based on the input image's gradient response, which is prone to false correspondence matching in a complex background. Recent edge-based methods [89, 60, 93, 94, 95, 96] that incorporate region color information demonstrate enhanced robustness.

Seo et al. [89] exploit local region color information of foreground and background to evaluate candidate image edge points. They search the object contour points only in the confident direction validated by region color information, rather than searching in both directions. Final corresponding points are edge points that maximize foreground and background separation. The local search is efficiently performed on the newly defined search bundle, which is a set of 1D search lines.

Seo et al. [89] effectively reduce false matches in cluttered background. However, their method independently selects each corresponding point and ignores the affinity of adjacent object contour points. Based on [89], Wang et al. [60] develop a graph model to describe the relationship between the candidate correspondences. Then the Global Optimal Search (GOS) of the correspondences is solved efficiently with dynamic programming. Compared with local optimal search in [89], the global optimal search (GOS) strategy captures the inherent property that the correspondence points for the contour of the object are essentially continuous and these points construct the object contour once connected. Consequently, this method proves more effective for complex 3D object models and highly cluttered backgrounds.

The region color information in [89, 60] is modelled by a single global foreground and background color histogram, which is insufficient to describe complex scenes. The searching strategy and pose optimization scheme are also prone to fail when the object is partially occluded. Inspired by region-based methods [98, 37], Huang et al. [93] use local color histograms to better model the region color information. During the correspondence searching, candidate points are first evaluated with edge color to filter out false contour points. Subsequently, each remaining candidate point is assigned a confidence value, measured by the region color and spatial distance. The optimal object contour points are searched by maximizing the edge confidence value. For pose optimization, the edge confidence value is used to weight the energy terms at each object contour point, thereby mitigating the influence of false correspondence matching. By incorporating the newly defined edge confidence, both the object contour point search and pose optimization are more robust to cluttered backgrounds and partial occlusions.

Li et al. [95] propose an optimized approach to fuse edge and region features by grouping them into a set of local bundles. The overall energy function is a weighted combination of these local bundles, with their weights dynamically adjusted based on the confidence values of the involved features. The spatially varying weighting of each bundle ensures better adaptation to different regions within the same frame. The energy function of each bundle comprises a weighted combination of the edge-based energy function from [92] and the region-based energy function in RBOT [37]. The features within each bundle are competitively weighted to optimize their complementarity.

In another study by Li et al. [96], a two-step optimization strategy is employed to leverage the speed advantages of edge-based methods and the accuracy advantages of region-based methods. Initially, a coarse pose is quickly estimated by optimizing the edge-based energy function proposed in [92]. Subsequently, the pose is refined with precision by optimizing the region-based energy function from [37]. In both edge-based and region-based approaches, a

reprojection process is typically required after each iteration to update the edge points or foreground points for optimization, which incurs high computational costs. To enhance efficiency, Li et al. [96] utilize the object's geometric properties to avoid the costly reprojection process in edge-based optimization and reduce the reprojection process to only once during region-based optimization. Specifically, the 3D foreground points are computed in the first reprojection process. For subsequent iterations, these points are updated using spatial coordinate transformation instead of relying on reprojection. By reducing the reprojection process to only once, significant efficiency improvements are achieved while maintaining accuracy.

Sun et al. [94] propose an edge-based method that leverages contours in a new way to enhance pose tracking robustness for less-distinct objects. Instead of sampling the contour into a sparse point set, they build a contour part model by segmenting the projected contour into distinguishable parts of certain lengths adaptively. The correspondence between the projected contour part and contour part in the query image is determined through 2D searching using the gradient orientation-based template matching. They also combine edge and region feature using a simple mode-switching strategy. The proposed method and the region-based method [37] run in parallel during the fusion process. For each frame, the pose with smaller updates is selected as the final result based on the hypothesis that the object moves continuously in the real world. This combined method outperforms both individual edge-based and region-based methods by utilizing robust contour features and sufficient region features.

4.1.2.4. Advanced optimization. Local optimization methods, such as gradient descent and Newton's method, can easily become trapped in local minima, particularly in complex, non-convex optimization problems. To overcome this issue and enhance tracking robustness, some studies propose combining local optimization techniques with global search strategies.

Bugaev et al. [91] combine keypoint-based tracking and contour energy optimization for monocular model-based 3D tracking. The initial object pose is estimated using the keypoint-based Kanade–Lucas–Tomasi (KLT) tracker. The pose is then refined by optimizing the contour energy function, which considers both the intensity and orientation of the model and image edges. For global optimization and to avoid local optima, the Basin-Hopping algorithm, combined with an efficient Quasi-Newton method [99], is used. The search area for the optimal energy function value is constrained using information from the preliminary pose estimated using the keypoint-based tracker. Combining keypoint-based and edge-based approaches can mitigate the typical disadvantages of both methods. However, due to its high computational complexity, this method is not suitable for real-time applications.

Tian et al. [97] propose a fast and effective tracker that combines contour-based local tracking with non-local searching to deal with large inter-frame displacements. They improve upon the contour-based tracking method [89, 93] by using long search lines, which are independent of pose update and can be precomputed. The precomputed searching line facilitates efficient non-local search, since hundreds of pose update iterations can be performed in real-time. To further enhance efficiency, non-local search is employed only for out-of-plane rotation, which requires sampling only in a 2D space instead of the original 6D pose space. The proposed method achieves significant improvements in handling large displacements while maintaining real-time performance.

4.1.2.5. Multiple pose hypotheses. The methods in section 4.1.2.2 that include multiple low-level hypotheses in robust registration have significantly improved tracker performance, but they still maintain a single hypothesis for the camera pose. Another category of approaches maintain multiple pose hypotheses using particle filtering. These methods predict poses based on probabilistic distributions without using 3D-2D correspondences, making them less sensitive to false contour edges. However, the computational cost is usually too high for real-time tracking since these methods need to evaluate large state spaces. These methods fall under the category of probabilistic tracking, which will be elaborated in section 4.3.1.

4.1.3. Region-based

Region-based methods jointly estimate the object pose and its foreground/background segmentation, enforcing alignment between the rendered silhouette of the 3D model and the object region observed in the image. Compared to keypoint-based or edge-based optimization, region-based methods are typically more robust in cluttered scenes and under moderate occlusion. As summarized in Table 4, these methods can be grouped by how region evidence is represented and evaluated: dense global models, dense local models, and sparse proxies. The table also reports the publication year, input type, optimization strategy, real-time capability, and the unique feature of each method,

Table 4: Overview of region-based instance-level 6D object pose tracking methods. Methods are grouped by how region evidence is represented (Dense Global, Dense Local, Sparse Proxies). Additional columns summarize publication year, input type, optimization strategy, real-time capability, and the unique feature.

Category	Method	Year	Input	Optimization	Real-time	Unique feature
Dense Global	PWP3D [36]	2012	RGB	Gradient descent	✓	First real-time region-based tracker
	Prisacariu [100]	2014	RGB	Levenberg-Marquardt	✓	Narrow-band SDF + hierarchical rendering
	Zhao [101]	2014	RGB	Damped Newton	✗	Occlusion-aware histogram update
	Tjaden [102]	2016	RGB	Gauss-Newton-like	✗	Robust optimization under fast rotations
Dense Local	2D-3D [103]	2016	RGB	Levenberg-Marquardt	✓	Localized FG/BG histograms around contour
	TCLC [98]	2017	RGB	Gauss-Newton-like	✓	Vertex-tied temporal consistent localized color histograms
	RBOT [37]	2018	RGB	Gauss-Newton	✓	Analytical derivatives for faster convergence
	Zhong [104]	2020	RGB	Gauss-Newton-like	✓	Polar sector partitioning with edge- and color-based occlusion reweighting
Sparse Proxies	Kehl [66]	2017	RGB-D	Gauss-Newton-like/Gauss-Newton	✓	Sparse contour sampling with depth fusion
	Huang [105]	2021	RGB	Gauss-Newton-like	✓	Search-line contour evaluation with reliability weighting
	SRT3D [38]	2022	RGB	Newton optimization + Tikhonov regularization	✓	Probabilistic correspondence lines
	ICG [67]	2022	RGB-D	Newton optimization + Tikhonov regularization	✓	Region-depth fusion with depth-based occlusion rejection
	ICG+ [106]	2023	RGB-D	Newton optimization + Tikhonov regularization	✓	Multi-modality with texture and multi-region cues
	FSCGE [63]	2024	RGB	Newton optimization + Tikhonov regularization	✓	Multi-feature fusion with edge correspondence lines
	Chen [107]	2025	RGB	Newton optimization + Tikhonov regularization	✓	Fan-shaped contour search with interior correspondences

highlighting their main contributions. The overall trend is clear: methods evolve from global color histograms to localized models and then to sparse correspondence proxies, progressively increasing efficiency and robustness. More recent work extends this progression by integrating multiple complementary cues such as depth, texture, edges, and interior correspondences into unified probabilistic frameworks, marking a shift toward multimodal fusion.

In the following, we review each category in detail. We begin with dense global models, then move to dense local models that introduce spatial adaptivity, and finally discuss sparse proxy methods that reformulate the problem into efficient one-dimensional evidence sources.

As summarized in Table 4, region-based methods can be grouped into three categories according to how region evidence is represented. Dense global models employ a single foreground and background appearance model, often based on global color histograms, which is optimized over a narrow band around the silhouette. These methods are simple and effective in controlled settings but are easily corrupted in clutter or when objects exhibit heterogeneous surfaces. Dense local models address this limitation by replacing global histograms with multiple localized models anchored to contour segments. By capturing local appearance variations, they offer improved robustness in complex environments, though at the cost of higher computational complexity. More recently, sparse proxy methods reformulate the problem by reducing dense two-dimensional evaluation to efficient one-dimensional evidence sources, such as rays or correspondence lines projected from 3D contour points. This not only improves computational efficiency but also facilitates principled handling of uncertainty and occlusion. In the following sections, we review the distinctive contributions of representative methods within each category.

4.1.3.1. Dense global models. PWP3D [36] establishes the foundation of region-based tracking from RGB images. Following the assumption that the 6D pose of the object corresponds to the optimal segmentation of the image into foreground and background, PWP3D defines an energy function that can be interpreted as the posterior of each pixel's foreground/background membership. A 3D model is rendered under the current pose, and its silhouette contour C is used to compute a signed distance field Φ, with the zero level-set defining the contour. For each pixel x in the 2D region Ω around the contour, the posterior probability is $P(\Phi|I) = \prod_{x\in\Omega}\Big(H_\Phi(x)P_f + (1 - H_\Phi(x))P_b\Big)$, where H_Φ is a smoothed Heaviside step function, and P_f, P_b are foreground/background posteriors derived from normalized RGB histograms updated online. Assuming pixel-wise independence and taking the negative log of $P(\Phi|I)$ yields

$$E = -\sum_{x\in\Omega} log\Big(H_\Phi(x)P_f + (1 - H_\Phi(x))P_b\Big). \tag{8}$$

Gradients are derived analytically, and pose updates are performed by gradient descent. Real-time performance is achieved through GPU parallelization. However, PWP3D relies on fixed step sizes that must be tuned per object and uses a fixed number of iteration, which can lead to convergence issues.

Prisacariu et al. [100] improve both optimization and efficiency by employing a hierarchical rendering approach combined with the second-order Levenberg-Marquardt algorithm to optimize pose parameters in a coarse-to-fine

manner. They also introduce a formulation that avoids the costly computation of the global signed distance transform and its derivatives by approximating the SDF derivatives only in a narrow band around the contour. This strategy accelerates convergence and enables real-time tracking even on mobile CPUs.

Zhao et al. [101] address both optimization and occlusion handling. They incorporate a particle filter-like stochastic optimization step into the pipeline, which provides better initialization for a subsequent damped Newton optimizer and helps avoid local minima. To mitigate histogram corruption under occlusion, they further propose an occlusion-aware online update strategy in which global color histograms are updated only if most of the pixels inside the projected contour belong to the foreground. This improves appearance model stability, but the occlusion handling applies only to histogram updates, not to pose optimization.

Tjaden et al. [102] focus on robust optimization and occlusion detection. They replace PWP3D's first-order gradient descent with a Gauss–Newton-like scheme, approximating the Hessian from first-order derivatives based on linearized twist parametrization. This allows automatic step-size selection and yields better convergence under fast rotations. In addition, they propose detecting mutual occlusion between objects by checking depth consistency at contour points, though this strategy requires knowledge of the occluding object's pose and model.

4.1.3.2. Dense local models. The global color histogram used in PWP3D is not descriptive enough for complex environments, such as cluttered backgrounds and heterogeneous objects. To address this limitation, Hexner and Hagege present a localized segmentation model in 2D–3D [103], in which multiple local circular regions are defined around the 2D object contour points. Each local region maintains its own foreground and background color models, and a local energy function measures the segmentation quality within that region. The overall energy is obtained by fusing all local energies, and pose parameters are optimized by minimizing this joint energy function.

The localized formulation of [103] improves segmentation performance in heterogeneous objects and cluttered environments. However, when the local background color closely resembles that of the object surface, each local region still suffers from the same ambiguity as global histograms. To improve the reliability of per-pixel energy terms, Tjaden et al. [98] compute average pixel-wise posterior probabilities from overlapping local regions instead of averaging energies, which leads to more accurate segmentation. To further ensure temporal consistency of the local color histograms, each region is associated with a 3D model vertex, enabling consistent identification and update across frames.

In a follow-up study, Tjaden et al. [37] retain this temporally consistent local histogram formulation but reformulate the Gauss-Newton optimization as an iteratively reweighted non-linear least-squares problem. The availability of analytical derivatives improves convergence speed and enhances robustness compared to their earlier Gauss-Newton-like scheme [102].

Zhong et al. [104] also propose a temporally consistent local model, but instead of linking regions to individual model vertices, they partition the global region into overlapping fan-shaped sectors identified by polar coordinates relative to the object center. This simplification requires fewer local regions to be maintained, while still ensuring temporal consistency and achieving comparable or superior segmentation results. To improve robustness under occlusion, [104] further introduce a weighting scheme based on edge distance and color cues. Sampling pixels far from image edges are assumed to correspond to occluded contour points, while color mismatches between rendered and observed edges provide an additional indicator of occlusion. Although effective, this strategy requires textured 3D models for rendering and may degrade under complex lighting conditions.

4.1.3.3. Sparse proxies. A major advance in real-time 6D object pose tracking is the transition from computationally expensive dense energy evaluations to efficient sparse frameworks. Instead of processing every pixel around the object silhouette, these methods evaluate evidence only along strategically selected rays or lines projected from 3D model points.

Early work by Kehl et al. [66] demonstrates the benefits of calculating the energy function sparsely along rays. Specifically, the 2D signed distance field (SDF) Φ is computed along rays originating from each projected contour point, with both positive and negative rays shot along the contour normal. To improve segmentation when the background is similar in color to the object, the contour-based energy term is combined with a geometry-based ICP term derived from depth data, with color posteriors reweighted according to the likelihood that a depth point belongs to the object. These complementary cues allow the system to compensate when color statistics are unreliable or depth data are noisy.

In Huang et al. [105], search lines sampled perpendicular to the contour are used to evaluate FG/BG posteriors of adjacent pixels, yielding a likelihood for each candidate contour point. The resulting likelihood serves not only to localize the contour but also to indicate whether a region is corrupted by occlusion or color ambiguity. If the likelihood is low, the weights of all pixels on the corresponding search line are reduced, preventing corrupted regions from dominating the optimization.

The sparse paradigm is formalized by Stoiber et al. [38] with SRT3D, which reformulates silhouette alignment as a probabilistic estimation problem over a set of correspondence lines. Each line, projected from a 3D contour point of a pre-rendered viewpoint model, defines an independent 1D segmentation problem. By assuming independence, line-wise probabilities are aggregated into a joint posterior over the object pose. This formulation achieves efficiency by avoiding dense distance transforms and online rendering, enabling real-time CPU performance. It also improves robustness by explicitly modeling local and global uncertainty, down-weighting occluded or ambiguous lines, and mitigating motion blur through aggregation across many lines. Optimization is carried out with second-order Newton updates and Tikhonov regularization, which stabilizes convergence by incorporating a prior on the previous pose.

ICG [67] extends SRT3D by fusing region and depth modalities in a joint probabilistic formulation. The region modality builds on SRT3D's correspondence lines, while the depth modality introduces correspondence points derived from depth maps using a point-to-plane metric. ICG also proposes an occlusion handling strategy: correspondences are validated against the measured depth map, and those whose expected depth conflicts with the observed depth are rejected. This prevents occluded or clutter-induced correspondences from biasing the optimization, allowing reliable cues to dominate.

ICG+ [106] extends the ICG framework by incorporating visual appearance cues in addition to geometry, resulting in a fully multi-modality tracker. Alongside region and depth information, it introduces two new modalities. The first is a texture modality based on 2D-3D keypoint correspondences, which establishes reliable matches to pre-stored keyframes and resolves geometric ambiguities in symmetric objects. The second is a multi-region modality, which partitions the object surface into patches rather than treating it as a single silhouette, allowing the tracker to exploit interior appearance cues when the outer contour is occluded or visually ambiguous. All modalities, including region, depth, texture, and multi-region, are fused within a unified probabilistic framework, yielding a robust system that achieves state-of-the-art accuracy in challenging conditions while maintaining efficient, real-time CPU performance.

FSCGE [63] builds on SRT3D's correspondence-line framework and introduces a multi-feature fusion strategy tailored for industrial object tracking under visual ambiguities. It combines an improved statistical color model, which integrates global and local regions to enhance robustness in complex scenes, with a geometric edge correspondence line (ECL) model that leverages gradient magnitude, orientation, and distance confidence of visible edges. This fusion enables the tracker to better handle symmetry, occlusion, and foreground-background similarity. To ensure complementary use of features, FSCGE employs an adaptive weighting mechanism that adjusts the contributions of color and edge cues based on real-time observation quality.

To address challenges in 6D object pose tracking for applications like Augmented Reality assembly, Chen et al. [107] extend the correspondence-line paradigm with several improvements. First, they introduce a fan-shaped search strategy, where multiple rays are projected around the contour normal to expand the search space, improving robustness against pose drift. Within this search, candidate contour points are modeled using a Gaussian–uniform mixture distribution, which explicitly accounts for both shape uncertainty and random noise to provide principled outlier suppression. Second, they add an interior correspondence modality, in which sparse interior points are pre-sampled from viewpoint templates and tracked with optical flow. These interior cues provide strong guidance for symmetric objects and large displacements, complementing contour evidence.

4.1.4. Direct optimization

Direct methods exploit dense raw pixel information in an image for pose tracking. Direct methods can be traced back to the famous direct image alignment framework: Lucas–Kanade (LK) framework [75], which is originally used to estimate 2D image transformations. Building upon the LK framework [75], direct methods have been widely used in planar tracking [108, 109], visual odometry [110] and 3D object tracking [39, 40].

The key assumption of direct methods is photometric constancy, which assumes that corresponding pixels between consecutive frames have the same pixel value. Pose parameters are optimized by densely aligning a template, in which the object pose is known, with the current image over a 3D object model, aiming to minimize the photometric error of corresponding object foreground pixels.

In order to apply the LK framework [75, 111] to 3D object tracking, a 3D warping function $W(\mathbf{x}, \boldsymbol{\mu}_T, \boldsymbol{\mu})$ is required to associate pixels between the template and the current image. This function transfers a point from the template image to a corresponding point in the current image by back-projecting $\mathbf{x}$ to the 3D model in pose $\boldsymbol{\mu}_T$ (the object pose for template $\mathbf{T}$), and then re-projecting it to the image plane in pose $\boldsymbol{\mu}$ (the object pose for current frame $\mathbf{I}$). The object pose $\boldsymbol{\mu}$ is found by minimizing the energy function, which measures the quality of the alignment between the template and the current image. The energy function is formulated as:

$$E(\boldsymbol{\mu}) = \sum_{\mathbf{x}} \| F(\mathbf{I}, W(\mathbf{x}, \boldsymbol{\mu}_T, \boldsymbol{\mu})) - F(\mathbf{T}, \mathbf{x}) \|^2 \tag{9}$$

where $F(\mathbf{J}, \mathbf{x})$ denotes a function that returns a descriptor for location $\mathbf{x}$ in a generic image $\mathbf{J}$. A commonly used descriptor function is $F(\mathbf{J}, \mathbf{x}) = \mathbf{J}(\mathbf{x})$, which is just the intensity in image $\mathbf{J}$ at location $\mathbf{x}$. Several algorithms have been proposed to efficiently optimize functions in this form, including the LK algorithm [75], the Inverse Compositional Algorithm (ICA) [111], and the Efficient Second-order Minimization (ESM) [112].

Even though direct methods have been promising for pose tracking, their underlying assumption (i.e., brightness constancy) is often violated by illumination changes or surface reflectance properties [113]. Many different approaches have been proposed to tackle the violation of brightness constancy assumption and to improve the robustness of direct methods. These improvements can be classified into 3 categories: illumination modeling, robust dense image descriptors and robust similarity measures.

4.1.4.1. Illumination modeling. Some works have attempted to explicitly model the influence of illumination changes. For instance, Seo and Wuest [40] estimate a compensation parameter using the surface normal of the object under Lambertian assumption and incorporate this parameter in the objective function to model intensity variations. This modeling allows for a more accurate prediction of image intensity, which in turn helps in aligning consecutive frames over the 3D target object. Baker and Matthews [111] introduce a model of linear appearance variation within the direct image alignment framework. This model captures how objects' appearance in images changes linearly with variations in factors like lighting and viewpoint. By incorporating this linear variation, the alignment process can better accommodate changes in appearance between the reference and target images. In general, estimating illumination parameters relies on modeling assumptions, which are challenging to generalize and formulate accurately due to the diverse and dynamic nature of real-world lighting conditions.

4.1.4.2. Robust dense image descriptors. Image intensity has been widely used as the image descriptor in direct methods. However, using image intensity itself as the descriptor is very sensitive to complex light changes, especially in the absence of texture. Some methods [39, 113, 109, 108] propose to use robust dense image descriptors instead of raw pixel values to improve the tracking performance with poorly textured objects, specular objects and lighting variations.

Crivellaro and Lepetit [39] propose gradient-based Descriptor Fields (DFs) [114]. Initially, convolutional filters are applied to the input image to extract local features, forming the basis for descriptor computation. Then a non-linear transformation is applied to the responses obtained from convolution filters, which separate the responses into positive and negative values. This segregation not only enhances the discriminative capability of these descriptors but also ensures robust alignment and broadens the basins of convergence.

Zhong et al. [113] build upon [39] to formulate a generic representation of dense image features termed as Extended Dense Feature Fields (EDFF), which is constructed using multiple pixel-level image features. Due to the complementary nature of different kinds of image features, an EDFF involving multiple features often enhances the robustness of pose tracking. Additionally, they propose to leverage dynamic textured model rendering in place of a fixed static set of templates. This method generates templates in real-time, continuously updating them based on the last estimated pose and illumination, thus achieving a realistic rendering to best resemble the current frame. This approach significantly improves convergence rates and handling of partial occlusions.

Chen et al. [109] propose an illumination-insensitive method for planar object tracking by introducing gradient orientation (GO) as the dense image descriptor within the framework of Efficient Second-order Minimization (ESM). This method is designed to be insensitive to illumination changes. The original ESM formulas are adapted to accommodate multidimensional features, allowing the integration of GO with ESM. Alismail et al. [108] demonstrate the integration of illumination-invariant binary descriptors [115] in a multi-channel Lucas and Kanade framework for enhanced tracking performance in low light and sudden illumination changes.

4.1.4.3. Robust similarity measures. Another family of works proposes to use intrinsically robust similarity measures rather than the difference-based cost function such as the sum of squared distances (SSD). Tracking based on the sum of squared distances (SSD) assumes constant brightness, which is not robust to illumination variation and often requires a photometric model to mitigate illumination changes.

Caron et al. [116] propose to optimize pose by maximizing the Mutual Information (MI) between two images: the current real captured image and a synthetic view evolving as the pose is optimized. Mutual Information (MI) quantifies the shared information between the two images, peaking when the two images are aligned. MI has been shown to be robust to noise, specular reflections, and even to different modalities. Exploiting this measure enables this pose tracking method to accommodate various image types (real and synthetic) and effectively handle occlusions and illumination changes. However, employing MI significantly increases the computational cost. Scandaroli et al. [117] employ a simpler measure, normalized cross correlation (NCC), which is invariant to affine illumination changes, and also presents low computational cost.

4.1.5. Summary

Table 5: Comparison of optimization-based 6D object pose tracking methods

	Keypoint-based	**Edge-based**	**Region-based**	**Direct Optimization**
Utilized information	Keypoint correspondence	Edge correspondence	Statistical information of image regions	Raw pixel values
Utilized pixels	Pixels inside the contour	Pixels at the contour	Pixels around the contour	Pixels inside the contour
Good at tracking	Heterogeneous/well-textured objects	Homogeneous/texture-less objects	Homogeneous/texture-less objects	Heterogeneous/well-textured objects
Failure cases	Texture-less objects	Heterogeneous objects in complex background	Heterogeneous objects in complex background, Symmetrical objects	Illumination changes, texture-less objects

Table 5 summarizes the characteristics of the four kinds of methods from the optimization-based approach: keypoint-based, edge-based, region-based, and direct optimization. They utilize various types of image information, and each is suitable for a specific tracking scenario.

Keypoint-based methods leverage distinctive local features to perform accurate pose estimation, making them robust to changes in object appearance and background clutter. However, they may struggle with texture-less objects lacking distinctive keypoints on their surfaces. Edge-based methods utilize edge information for pose tracking, offering robustness to texture-less objects but may fail in low-contrast environments where edge detection is difficult or cluttered backgrounds where strong edges from the background might cause false edge matching. Region-based methods analyze pixel-wise color statistics around the object contour, enabling robust tracking of a wide range of objects in cluttered scenes. However, they might still fail in complex backgrounds where the object is hardly distinguishable from the background. Moreover, region-based methods suffer from pose ambiguities because they determine the 3D object pose only from 2D silhouette information, making tracking of symmetrical objects prone to errors. Instead of extracting local features such as keypoints or edge-based methods, direct methods utilize rich information in an image to contribute to the pose tracking. Like keypoint-based methods, direct methods favor well-textured objects. They may fail for purely texture-less objects or under significant illumination changes. Each method has its strengths and weaknesses, and the choice depends on the specific characteristics of the object and the tracking scenario. The complementary properties of these methods also allow for the combination of various techniques to achieve more robust tracking.

4.2. Learning-based approach

When an object moves from pose $\mathbf{T}_{k-1}$ to pose $\mathbf{T}_k$, a discrepancy arises between the expected object data at pose $\mathbf{T}_{k-1}$ and the observed data $\mathbf{I}_k$ formulated by the actual current pose. Optimization-based methods define an energy function to quantify the discrepancy and solve the pose parameters by minimizing the energy function. In contrast, learning-based methods train a regressor to learn the relationship between the discrepancy and the relative transform $\Delta\mathbf{T}$ that takes the object from $\mathbf{T}_{k-1}$ to the pose $\mathbf{T}_k$ captured by the current observation.

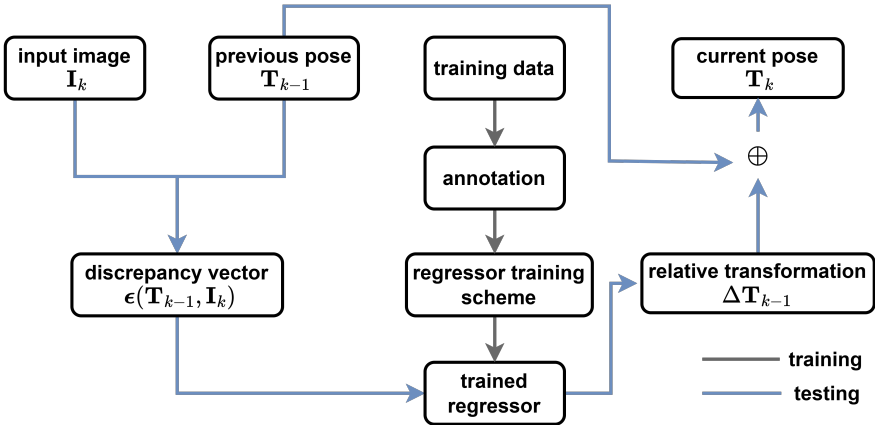


Figure 4: Overall schematic representation of learning-based methods.

Fig.4 illustrates the schematic of learning-based approaches, which formulate 6D object pose tracking as a regression problem. The underlying principle of learning involves introducing perturbations to the ground truth pose, simulating the transformation from the previous timestamp. During training, the perturbation pose parameter vector $\Delta\boldsymbol{\mu}$ is randomly sampled, and the resulting transformation $\Delta\mathbf{T} = \mathbf{T}(\Delta\boldsymbol{\mu})$ is applied to the ground truth object pose, yielding $\mathbf{T}_{noisy} = \Delta\mathbf{T} \cdot \mathbf{T}_{gt}$. Each transformation induces distinct effects on the discrepancy. The regressor is trained to correlate the impact of $\Delta\boldsymbol{\mu}$ on the discrepancy vector $\boldsymbol{\epsilon}(\mathbf{T}_{noisy}, \mathbf{I}_{gt})$. During testing, given the previous pose $\mathbf{T}_{k-1}$ and the current observation $\mathbf{I}_k$, the trained regressor utilizes $\boldsymbol{\epsilon}(\mathbf{T}_{k-1}, \mathbf{I}_t)$ to predict $\Delta\boldsymbol{\mu}_{k-1}$ and updates $\mathbf{T}_{k-1}$ to derive the current pose $\mathbf{T}_k = \Delta\mathbf{T}_{k-1} \cdot \mathbf{T}_{k-1}$.

Early learning-based approaches train the regressor from depth data using random forests with handcrafted discrepancy features. Recent methods leverage Deep Neural Networks (DNNs) to implicitly extract discrepancy features and directly learn tracking from image data. In the following sections, we delve into the details of random-forest-based and deep-learning-based 6D object pose trackers.

4.2.1. Random-forest-based

Random Forest Regression leverages multiple decision trees to make accurate and robust predictions. During training, each tree is trained independently on a random subset of the dataset, introducing diversity. At each node, a random subset of features is considered, and the best feature and threshold are selected to split the data into two subsets. The recursive splitting stops when a stopping criterion is met, such as reaching maximum depth or having a minimum number of samples in a node. When the criterion is reached, the node becomes a leaf node, storing a summary statistic like the mean and standard deviation of target variable values for the samples reaching it. During testing, input samples traverse from the root to leaf nodes using the splitting parameters, and predictions from all trees are aggregated to form the final prediction. This ensemble approach mitigates overfitting and enhances prediction accuracy, effectively handling challenges like noise and missing data.

To leverage random forest regression for predicting $\Delta\boldsymbol{\mu}$ in 6D pose tracking, the first step involves creating a training dataset. Depth images are generated by rendering the object's CAD model from n_c different viewpoints to capture comprehensive coverage of the target object. This process yields diverse $\mathbf{T}_{gt}$ poses and corresponding images $\mathbf{I}_{gt}$. Next, pose shifts $\Delta\mathbf{T}$ and their corresponding noisy poses $\mathbf{T}_{noisy}$ are formulated by randomly sampling n_w pose parameter vectors $\Delta\boldsymbol{\mu}$. To extract the discrepancy vector, n_m pairs of corresponding points between the object model and the depth data are established through projective association. The scene points from the depth image are transformed to the object coordinate system using $\mathbf{T}_{noisy}$. The signed displacement between a transformed point and its corresponding point on the object model M characterizes the discrepancy. The individual discrepancy values from all sampled points collectively form a discrepancy vector $\boldsymbol{\epsilon}(\mathbf{T}_{noisy}, \mathbf{I}_{gt}) \in \mathbb{R}^{n_m}$, serving as one training sample input. Each pose parameter in $\Delta\boldsymbol{\mu}$ is treated independently, resulting in the creation of one forest per parameter. The target training output corresponds to one of the n_p pose parameters in $\Delta\boldsymbol{\mu}$.

The first learning-based method for 6D pose tracking using depth images was introduced in [64]. This approach

employs a point sampling strategy restricted to one view of the object, with each camera view having its own set of sample points. For each of the n_c views, one forest consisting of n_t trees is trained. The 6D pose is represented by three translation components and three Euler angles, resulting in a total of $6 \cdot n_c \cdot n_t$ trained trees. During tracking, the closest learned camera view to the current camera position is used for prediction. From the current depth image $\mathbf{I}_k$, the visible surface of the object model is enclosed by an $n_g \times n_g$ regular grid, where the points belonging to the object are back-projected and used as the sample points. Then the discrepancy vector for those points $\boldsymbol{\epsilon}(\mathbf{T}_{k-1}, \mathbf{I}_k)$ is calculated and passed to all trees, yielding n_t predictions. A percentage of the n_t predictions with the lowest learned standard deviation are aggregated and their means are averaged to produce the final prediction for one dimension in $\Delta\boldsymbol{\mu}$. Once all 6 dimensions of $\Delta\boldsymbol{\mu}$ are predicted, $\Delta\boldsymbol{\mu}$ is used to update the previous pose $\mathbf{T}_{k-1}$ and derive the current pose $\mathbf{T}_k$.

Inspired by [64], Joseph et al. [118] propose an approach with reduced computational cost and enhanced occlusion handling capabilities. A similar point sampling method as in Tan et al. [64] is utilized, and one forest is trained for each of the n_c camera views. However, in contrast to Tan et al. [64], which evaluates a number of trees from the same view for final prediction, Joseph et al. [118] utilize trees from multiple neighboring camera views for final prediction, resulting in a significant reduction in the number of trees per view to just one. This adaptation yields a total of $6 \cdot n_c$ trees in the forests across all views. Occlusion is handled by discarding a subregion of the object and selecting points exclusively from the remaining subregion for training trees.

Akkaladevi et al. [58] present a framework that unites object pose estimation and tracking to track multiple objects in 3D space. The adopted point sampling strategy is not restricted to a single view. Approximately 400 points from the object's surface are sampled by raytracing onto the object's surface, creating a 3D grid around the object, and then sampling a single point per grid. This method results in evenly spaced sample points covering the entire visible surface of the object. Given that points are sampled from all around the object, many points may not lie on the visible surface but rather be positioned behind it. The random forest can utilize this information to infer object movements. With this point sampling strategy, it has been shown that training only six random forests is sufficient for learning $\Delta\boldsymbol{\mu}$, resulting in a substantial reduction in computational complexity.

4.2.2. Deep-learning-based

The ensemble approach of a random forest proves effective in handling occlusion and noise in simple scenarios. However, its performance in more complex conditions remains unsatisfactory due to the hand-designed features and the limited learning capability. To overcome these challenges, recent studies have introduced deep neural networks to learn the relative pose between two frames. These networks implicitly learn to calibrate the discrepancy between features extracted from the current observation and the rendered image conditioned on the previous pose estimate, enabling them to predict the relative transform more effectively in complicated conditions.

Overall Scheme Figure 5 illustrates the overall framework of deep learning-based 6D pose tracking methods. The network takes as input a pair of images: $\mathbf{I}_{pred}$, a rendered image of the object generated using the CAD model and the pose from the previous frame $\mathbf{T}_{k-1}$, and $\mathbf{I}_{obs}$, a cropped and resized observation of the object from the current frame $\mathbf{I}_k$. The cropping is guided by a bounding box projected from the previous pose estimate. These two images are passed through a trained relative pose estimation network. First, features are extracted from both $\mathbf{I}_{pred}$ and $\mathbf{I}_{obs}$, followed by a feature fusion stage to integrate information from both inputs. The fused features are then processed by the pose prediction head to estimate the relative pose $\Delta\mathbf{T}_{k-1}$ between the two frames. Finally, the current pose $\mathbf{T}_k$ is computed by applying the relative pose update to the previous pose.

Training In the training phase, images annotated with ground truth 6D object poses are assumed known and can be used as the observed input. However, real training images from existing datasets may lack data in certain situations such as occlusions between objects or variations in lighting. To ensure the network's robustness in diverse scenarios, most methods employ synthetic datasets for training the neural network.

The observed image $\mathbf{I}_{obs}$ is generated by randomly sampling a pose $\mathbf{T}_{obs}$ and rendering the object CAD model onto a randomly chosen real background image under varying lighting conditions. This process aims to mimic diverse backgrounds and lighting conditions encountered in real testing scenarios. Subsequently, a perturbation pose $\Delta\mathbf{T}$ is randomly sampled and applied to $\mathbf{T}_{obs}$ to derive $\mathbf{T}_{pred}$. The rendered image $\mathbf{I}_{pred}$ is generated by rendering the object model onto a blank background under a fixed lighting condition using the noisy pose $\mathbf{T}_{pred}$. This rendering process eliminates cluttered backgrounds, focusing solely on the object's area, which allows the neural network to extract more meaningful features.

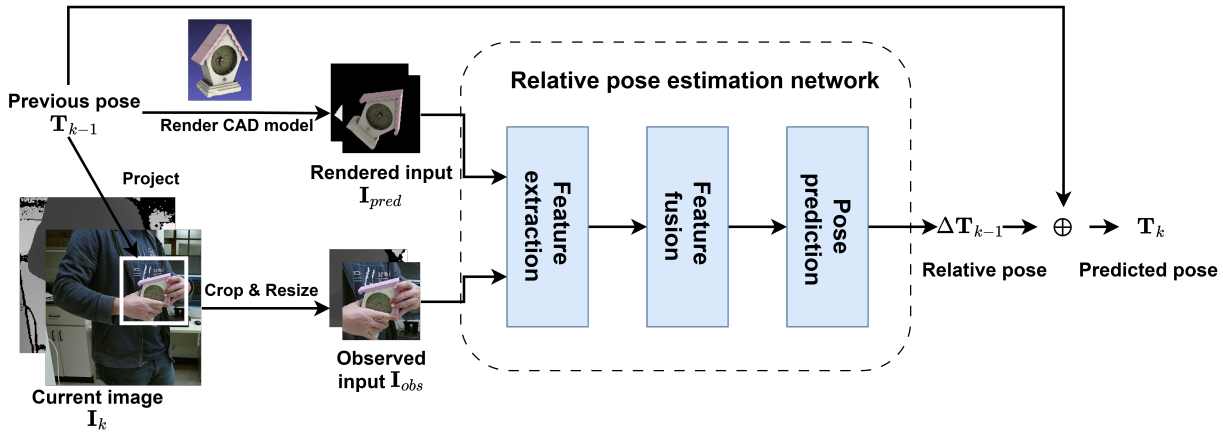


Figure 5: Overview of deep-learning-based methods.

As the network expects fixed size inputs, raw rendered and observed images are firstly cropped using an enlarged bounding box that bounds the object's 3D model, projected onto the image space using the predicted pose $\mathbf{T}_{pred}$. After cropping, the images are resized to the target resolution. The network is trained by minimizing a loss function defined over the target and estimated pose change parameter $\Delta\boldsymbol{\mu}$, which is used to parameterize $\Delta\mathbf{T}$.

When generating a training dataset for 6D pose tracking, several key considerations should be taken into account. Firstly, the observed pose $\mathbf{T}_{obs}$ should encompass sufficient viewpoint coverage, and the images should be rendered at varying scales depending on the range within which the target object is situated. Secondly, it is essential to choose a proper distribution from which perturbation poses are sampled. Typically, the translation and rotation directions are uniformly sampled, while the magnitude of translation and rotation follow either a Gaussian distribution or Uniform distribution. Thirdly, reducing the simulation-to-reality gap in synthetic datasets is crucial for robust 6D pose tracking in real-world testing scenarios. During training, both observed and rendered inputs are synthetically generated. However, during tracking, the observed image comes from a real sensor. To reduce the simulation-to-reality gap and improve the robustness of the trained networks, various domain randomization techniques can be employed while rendering the observed images during training. These techniques include random HSV shift, Gaussian noise, Gaussian blur, occlusion and depth-missing corruption.

In this section, we provide a detailed review of deep-learning-based instance-level 6D pose tracking methods. Among these, DeepIM [119] and DMB6D [120] are pose refinement techniques designed to improve the accuracy of poses initially estimated by another method. However, they can also function as pose tracking methods by refining the previous pose estimation iteratively. Additionally, DPOD [122], Trabelsi et al. [123], DenseFusion [121], and FoundationPose [23] are pose estimation methods that incorporate a pose refinement module. In this review, we focus exclusively on their pose refinement components. Although these methods adhere to a common paradigm, they differ primarily in network design, including feature extraction and fusion strategies, prediction heads, and loss functions. To provide a structured analysis, we categorize our review based on these key design aspects, highlighting the unique architectural choices made by each approach. Finally, we summarize and compare these methods in Table 6.

4.2.2.1. Feature extraction and fusion. Deep 6-DOF Tracking (D6DT) [41] was the first approach to leverage deep learning for instance-level object pose tracking using RGB-D sequences. The network processes the predicted and observed frames independently through a single convolution layer with separate weights. The extracted features are then concatenated and passed through three additional convolution layers for high-level feature extraction.

A subsequent work [42] introduced architectural improvements by replacing the convolution layers with "fire" modules from SqueezeNet [127]. These modules employ squeeze-and-expand operations, significantly reducing the number of parameters while maintaining expressiveness and improving generalization. Additionally, the network concatenates input features at each layer with their corresponding outputs before max-pooling, preserving spatial and hierarchical information across layers. This modification enhances the model's ability to learn robust representations, mitigating the domain gap between synthetic and real data. As a result, the model generalizes better to real-world

Table 6: Characteristics of the deep-learning-based instance-level 6D object pose tracking methods.

Method	Year	Input	Training data	Feature extractor	Rotation parameterization	Pose regression loss	Real-time capable?	Unique feature
D6DT [41]	2017	RGB-D	R&S	Custom CNN	Euler angle	L_2	✓	First deep learning-based 6D pose tracker
Garon [42]	2018	RGB-D	S	Custom CNN	Euler angle	L_2	✓	Improved generalization with "fire" module
DeepIM [119]	2018	RGB	R&S	FlowNet	quaternion	ADD	✗	Intermediate disentangled pose representation
DMB6D [120]	2018	RGB	R&S	InceptionV4	quaternion	contour	✓	New contour loss to handle visual ambiguity
DenseFusion [121]	2019	RGB-D	R&S	CNN, PointNet	quaternion	ADD	✗	Pixel-wise dense fusion of color image and point cloud embeddings
DPOD [122]	2019	RGB	R&S	ResNet	quaternion	ADD	✓	3 separate heads for XY,Z and rotation prediction
se(3)-TrackNet [43]	2020	RGB-D	S	Custom CNN	Lie algebra	L_2	✓	Effective 3D orientation representation via Lie Algebra
Trabelsi [123]	2021	RGB	R&S	FlowNet	quaternion	ADD	✓	Spatial Multi-Attentional Block
HFF6D [124]	2022	RGB-D	S	ResNet	Lie algebra	L_2	✓	Hierarchical feature fusion
FCR-TrackNet [125]	2023	RGB-D	S	Custom CNN	Axis angle	L_2	✓	Joint classification-regression prediction head
Wang [126]	2024	RGB	S	Custom CNN	Axis angle	L_2	✓	Cross domain attention fusion
FoundationPose [23]	2024	RGB-D	S	Transformer	Axis angle	L_2	✓	Large-scale synthetic training and transformer-based architecture

R: **R**eal data S: **S**ynthetic data

scenarios without requiring fine-tuning on real images, as in D6DT.

DeepIM [119] introduced the first iterative optimization network that operates exclusively on RGB images. Its architecture is built upon FlowNetSimple [128], an optical flow estimation network that predicts dense, pixel-wise motion between consecutive frames using a series of convolutional and deconvolutional layers. In DeepIM, the observed and rendered image pairs are concatenated along the channel dimension and passed through FlowNetSimple convolutional layers. The extracted feature map is then passed through fully connected layers for pose prediction. During training, the feature map is also fed into FlowNetSimple's deconvolutional layers to reconstruct optical flow and foreground masks as auxiliary supervision tasks.

DMB6D [120] is also designed to refine 6D pose estimates using only RGB images. It extracts low-level features from both the rendered and observed frames using the first 5 blocks of InceptionV4 [129]. These extracted low-level features are then concatenated and processed through a shared high-level feature encoder.

The pose refinement network of DPOD [122] employs two parallel input branches, each consisting of the first five ResNet [130] layers to extract features from the observed and rendered images. The extracted feature representations from both branches are then subtracted to capture differences between the two inputs. This difference is subsequently passed through an additional ResNet block, generating a refined feature for pose prediction.

The pose refinement module of [123], known as MARN (Multi-Attentional Refinement Network), uses a multi-attention mechanism to focus on the most discriminative object features while mitigating occlusion effects. The network utilizes a ResNet18 encoder to extract visual features and simultaneously estimates optical flow between rendered and observed frames using FlowNetSimple [128]. The extracted rendered features $\boldsymbol{F}_r$ are then warped using optical flow to better align with the observed image feature $\boldsymbol{F}_o$. To generate attention maps, the warped feature map $\boldsymbol{F}_w$ undergoes two 1×1 convolutional layers, producing a set of summarized feature maps, which are normalized via softmax to obtain multiple attention maps. Each attention map is then element-wise multiplied with the corresponding enhanced observed feature representation $\boldsymbol{F}_o^+$, which includes concatenated observed features and flow information, generating N separate attention-enhanced feature maps. These maps are concatenated to form the final multi-attentional feature representation, enabling the network to robustly localize discriminative parts of the target object. In cases of partial occlusion, the multi-attention module adaptively focuses on visible object parts while ignoring occluded regions, ensuring more reliable pose refinement.

DenseFusion [121] introduces a 6D pose estimation framework for known objects in RGB-D images, integrating

a main pose estimator and a pose residual estimator for improved accuracy. Instead of treating depth as an additional image channel in a CNN, DenseFusion converts depth pixels into a 3D point cloud using camera intrinsics and extracts geometric features via a PointNet-like architecture. Simultaneously, color image crops are processed using an encoder-decoder CNN, where each pixel embedding represents appearance information at the corresponding location. The network then fuses color and geometric features at corresponding pixels, ensuring precise spatial alignment. Each fused pixel feature independently predicts a 6D pose, and the final pose is obtained by aggregating these predictions. To further enhance accuracy, a pose residual estimator iteratively refines the pose by reusing color feature embeddings from the main network and performing dense fusion with updated geometric features extracted from the transformed point cloud based on the latest pose estimate. This iterative refinement significantly improves robustness in cluttered and occluded environments.

Unlike previous approaches that fuse rendered and observed frame features only at the final encoding layer, HFF6D [124] introduces a hierarchical feature fusion framework with Subtraction Feature Fusion (SFF) modules for richer spatial-temporal interaction. A ResNet-like feature extractor processes both frames independently through multiple encoding layers, extracting hierarchical representations. At each encoding level, the SFF module computes the difference between features, highlighting regions where motion or pose changes occur. This difference map is then passed through an attention mechanism, where the network learns adaptive weights to focus on the most relevant differences. The attention-weighted difference map is then concatenated with the original feature maps, enhancing the feature representation at each level. The final fused feature representation is obtained by concatenating hierarchical features from different encoder layers. The network progressively integrates feature information from different scales, promoting hierarchical feature interaction between the two input branches and enabling robust relative pose estimation.

As in HFF6D, FCR-TrackNet [125] fuses multi-level features from RGB-D inputs to enhance pose tracking accuracy. Its low-level feature extraction consists of two independent branches, each comprising a convolutional block, a max pooling layer, and a residual block. The extracted feature matrices from both branches are concatenated into a single unit before being fed into the high-level feature extraction network. This high-level network consists of two branches: one applies a convolutional block and a max pooling layer to extract stronger semantic information, while the other uses only a max pooling layer to retain low-level spatial details while reducing resolution. Finally, the low-level and high-level features are concatenated, ensuring that both the original physically rich details and high-level semantic information contribute to robust 6D pose tracking.

Wang et al. [126] proposed a Cross-Domain Attention Fusion Module to enhance feature matching between the observed and rendered images using self-attention mechanisms in both spatial and channel dimensions. The spatial attention module computes query $\boldsymbol{q}_o$, key $\boldsymbol{k}_r$, and value $\boldsymbol{v}_r$ vectors to establish pixel-level alignments between the two images. The observed image feature $\boldsymbol{F}_o$ is processed through a convolutional layer, then flattened along the spatial dimension to generate $\boldsymbol{q}_o$. The rendered image feature $\boldsymbol{F}_r$ undergoes the same transformation twice to produce $\boldsymbol{k}_r$, and $\boldsymbol{v}_r$. A dot-product similarity score between $\boldsymbol{q}_o$ and $\boldsymbol{k}_r$ generates an attention weight matrix, which is softmax-normalized. The attended feature map $\boldsymbol{F}_s$ is computed by weighting $\boldsymbol{v}_r$ with attention scores and is fused with $\boldsymbol{F}_o$ using learnable weights, preserving long-range dependencies. The channel attention module follows the same process but operates along the channel dimension instead, allowing the network to selectively enhance object-relevant features while filtering out background noise. Finally, the outputs from the spatial and channel attention modules are aggregated into a refined feature representation, which is passed into the pose residual estimation module. This approach improves pose tracking accuracy, particularly for texture-less objects and occlusion-heavy environments.

The pose refinement network in FoundationPose [23] enhances an initial coarse pose estimate by predicting incremental updates for translation and rotation. The rendered input and observed input first pass through a shared-weight CNN encoder, where extracted feature maps are concatenated and processed using CNN blocks with residual connections. The output is tokenized into patches with positional embeddings to retain spatial information and prepare for transformer-based processing. The network then predicts the translation update and rotation update, each individually processed by a transformer [131] encoder and linearly projected to the output dimension. This transformer-based architecture improves generalization, enabling the model to effectively perform pose tracking for novel objects that were not seen during training.

4.2.2.2. Prediction heads. The prediction heads take the fused feature from feature extraction block to estimate the relative pose transformation between the rendered image and the observed image, which is composed of relative

translation $\Delta \boldsymbol{t}$ and relative rotation $\Delta \boldsymbol{w}$. The relative translation $\Delta \boldsymbol{t} \in \mathbb{R}^3$ is expressed as $\Delta \boldsymbol{t} = \boldsymbol{t}_k - \boldsymbol{t}_{k-1}$, which represents the object's translation shift in the camera frame. The relative rotation $\Delta \boldsymbol{w}$ represents the object's orientation update expressed in the camera frame. Its formulation varies depending on the rotation representation used in different approaches. Below, we describe how $\Delta \boldsymbol{w}$ is formulated in various commonly used representations.

Euler angles represent the rotation using three consecutive rotations about specific axes. The relative rotation in Euler angles is computed as: $\Delta \boldsymbol{w} = \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}$, where $\boldsymbol{\theta} = (\theta_x, \theta_y, \theta_z)$ represents the roll, pitch, and yaw angles. However, this representation suffers from singularities, known as gimbal lock, which can lead to numerical instability in optimization-based pose estimation.

Quaternions provide a compact and singularity-free representation of 3D rotation using a four-dimensional vector $\boldsymbol{q} = (q_w, q_x, q_y, q_z)$. The relative rotation in quaternion form is computed using quaternion multiplication $\Delta \boldsymbol{w} = \boldsymbol{q}_k \otimes \boldsymbol{q}_{k-1}^{-1}$, where $\otimes$ denotes quaternion multiplication and $\boldsymbol{q}^{-1}$ is the inverse of a quaternion. This representation avoids singularities but requires additional constraints (such as normalization) to ensure unit norm.

In the axis-angle representation, rotation is described using a unit vector $\boldsymbol{v}$ representing the axis of rotation and an angle θ about that axis. The relative rotation is given by $\Delta \boldsymbol{w} = \theta \boldsymbol{v}$, where $\boldsymbol{v} \in \mathbb{R}^3$ is the unit rotation axis, and θ is the rotation magnitude. This representation provides an intuitive interpretation of rotation but suffer from ambiguities in representation, i.e, a rotation by an angle θ around an axis $\boldsymbol{v}$ produces the same effect as a rotation by $-\theta$ around $-\boldsymbol{v}$. This means that the same physical rotation can have two different mathematical representations, which can lead to inconsistencies when comparing rotations or interpolating between them.

The Lie algebra representation expresses the relative rotation as an element of the Lie algebra $\mathfrak{se}(3)$, where the rotational component is represented as a 3D vector in the tangent space of $SE(3)$. The relative rotation vector is given by $\Delta \boldsymbol{w} = \log(\mathbf{R}_k \mathbf{R}_{k-1}^T)$, where $\log(\cdot)$ denotes the matrix logarithm, and $\mathbf{R}_k$ represents the rotation matrix at time step k. This representation provides a compact and smooth way to model small rotational changes and is widely used in optimization-based motion estimation.

The most straightforward approach to predicting the relative pose transformation is to use a single prediction head for both relative translation and relative rotation, as seen in D6DT [41] and [42]. However, this entangles translation and rotation estimation, potentially leading to suboptimal predictions. More recent methods, such as SE3-TrackNet [43] and HFF6D [124], address this issue by disentangling the predictions of $\Delta \boldsymbol{w}$ (relative rotation) and $\Delta \boldsymbol{t}$ (relative translation) into separate branches. This separation enhances learning capability, stability, and overall prediction accuracy.

Due to the difficulty of directly regressing relative pose transformation from RGB images without depth information, DeepIM [119] introduces a disentangled representation to simplify the learning task. Instead of predicting absolute 3D translation, DeepIM estimates relative translation in image space, reducing the complexity of the solution space. The network regresses the relative translation $\Delta \boldsymbol{t} = (v_x, v_y, v_z)$, where v_x and v_y represent the shift in pixels along the image's x-axis and y-axis, respectively. v_z represents the scale change of the object, effectively capturing depth variations. For rotation representation, the center of rotation is set to the object's center in the camera frame, rather than the camera's origin. This prevents rotations from introducing unintended translations in the camera coordinate system. Furthermore, by aligning the rotational axes with the camera frame, DeepIM eliminates the need for object-specific coordinate transformations, making the representation independent of the object's actual size, its internal model coordinate system, and camera intrinsics. This approach improves generalization across different objects and simplifies training. Following DeepIM, DPOD [122] also adopts a disentangled representation for pose refinement. However, DPOD further splits translation prediction into two separate heads: One for XY translation in image space. Another for Z direction translation to account for their different scales.

FCR-TrackNet [125] introduces a joint classification-regression approach that narrows the range of rotation estimation, enhancing both convergence speed and accuracy in rotation prediction. The network learns the axis-angle rotation vector through a two-step process: coarse classification followed by fine-grained regression. First, the rotation angle θ is discretized into bins, and the network classifies it into one of these bins using Classification label for Rotation Angles (CRA). The CRA classifies the rotation angle into different ranges and converts the discrete angle space into a smooth coding label, therefore mitigates discontinuities at angle boundary. Next, a residual rotation $\delta\theta$ is regressed within the selected bin, refining the prediction for continuous rotation estimation. Many real-world objects exhibit rotational symmetry, where multiple different rotations produce identical visual appearances. Direct regression struggles with this ambiguity, forcing a single fixed solution even when multiple valid ones exist, leading to unstable training and inconsistent predictions. The joint classification-regression approach in FCR-TrackNet ensures stable

learning and higher accuracy, particularly for rotationally symmetric objects.

Beyond relative pose prediction, some methods incorporate auxiliary learning tasks to enhance performance. In DeepIM [119], feature maps used for pose transformation prediction are also leveraged for optical flow and foreground mask estimation. Similarly, Wang et al. [126] include a semantic segmentation head to predict the foreground mask of the observed image. These auxiliary tasks regularize feature representation, improve training stability, and enhance pose estimation accuracy.

4.2.2.3. Loss functions. The loss function for relative pose prediction models consists of pose regression loss and optional auxiliary losses, which help enhance stability, feature learning, and generalization. DeepIM [119] incorporates optical flow loss to minimize flow prediction errors and uses sigmoid cross-entropy loss as foreground mask loss. Wang et al. [126] add a semantic segmentation head to improve training stability, adopting softmax cross-entropy loss for semantic label training. MARN combines pose regression loss with a regularization term that enforces orthogonality among attention maps, preventing multiple attention maps from focusing on the same discriminative object part. Next, we introduce the pose regression losses commonly used in the literature.

L_2 **loss** The L_2 loss function is defined as:

$$L = \lambda_1 L_{rot} + \lambda_2 L_{trans} \tag{10}$$

$$L_{rot} = \|\Delta \boldsymbol{t} - \Delta \hat{\boldsymbol{t}}\|_2 \quad \& \quad L_{trans} = \|\Delta \boldsymbol{w} - \Delta \hat{\boldsymbol{w}}\|_2 \tag{11}$$

, where $\Delta \boldsymbol{t}$ and $\Delta \boldsymbol{w}$ denote the ground truth relative translation and rotation, while $\Delta \hat{\boldsymbol{t}}$ and $\Delta \hat{\boldsymbol{w}}$ are the predicted relative translation and rotation. λ_1 and λ_2 represent the weight parameters of each prediction component.

ADD loss The ADD loss, similar to the ADD metric (introduced in 2.2.4), measures the mean distance between corresponding 3D points transformed by the predicted pose and the ground truth pose:

$$L_{ADD} = \frac{1}{m} \sum_{\mathbf{x} \in M} \| (\mathbf{R}\mathbf{x} + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{t}}) \| \tag{12}$$

Here, x denotes the object's 3D points and m is the number of points. $\mathbf{R}, \mathbf{t}$ denote the ground truth rotation and translation, while $\hat{\mathbf{R}}, \hat{\mathbf{t}}$ are the predicted rotation and translation. The ADD loss is primarily used for asymmetric objects. To handle symmetric objects, ADD-S loss can be used, which modifies the distance computation to accommodate symmetry. Instead of computing the distance between corresponding 3D points directly, ADD-S loss determines the error by finding the minimum distance between each transformed point and the closest point in the target set:

$$L_{ADD-S} = \frac{1}{m} \sum_{\mathbf{x}_1 \in M} \min_{\mathbf{x}_2 \in M} \| (\mathbf{R}\mathbf{x}_1 + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x}_2 + \hat{\mathbf{t}}) \| \tag{13}$$

Another variation of ADD loss replaces the L_2 norm with the average L_1 distance to reduce sensitivity to outliers. The L_1 norm helps mitigate large graidents caused by outliers and maintain loss stability during training.

Contour loss DMB6D [120] proposed a contour tracking loss to handle the visual ambiguity due to symmetry or similar appearances from multiple viewpoints. Inspired by region-based methods, DMB6D transforms the complex problem of 3D geometric alignment into a simpler 2D silhouette matching task via a distance transform. The network learns to optimize 6D pose refinement by adjusting rotation and translation to maximize projected contour alignment.

For a given training sample with input pair (S, H), a distance transform of the observed scene contour D_S and hypothesis rendered contour points V_H, contour loss is defined as:

$$L(\Delta \boldsymbol{q}, \Delta \boldsymbol{t}, D_S, V_H) = \sum_{v \in V_H} D_S \left[\pi \left(\Delta \boldsymbol{q} \cdot v \cdot \Delta \boldsymbol{q}^{-1} + \Delta \boldsymbol{t} \right) \right] \tag{14}$$

with $\Delta \boldsymbol{q}^{-1}$ being the conjugate quaternion and $\Delta \boldsymbol{t}$ is the relative translation. The distance transform D_S of the scene image assigns a distance value to each pixel, indicating how far it is from the nearest edge. A bi-directional contour loss is introduced to further prevent local minima. In addition to aligning the hypothesis contour V_H to the scene distance transform D_S, the scene contour V_S is also aligned with the hypothesis distance transform D_H. The final contour loss is the sum of $L(\Delta \boldsymbol{q}, \Delta \boldsymbol{t}, D_S, V_H)$ and $L(\Delta \boldsymbol{q}^{-1}, -\Delta \boldsymbol{t}, D_H, V_S)$.

4.2.2.4. Summary. Table 6 provides an overview of deep-learning-based instance-level 6D pose tracking methods. These networks take a rendered image and an observed image as input, processing them through either a shared CNN encoder or separate CNN encoders. The extracted features are then fused and passed through additional layers for high-level feature extraction, which is subsequently used to predict relative translation and rotation updates. Several key design choices influence network performance, including the feature extractor, feature fusion strategy, output representation, and training loss.

Methods that leverage powerful feature extractors such as transformers exhibit strong generalization capability. Additionally, multi-level feature fusion and enhanced interaction between observed and rendered features improve the extraction of discriminative features for pose refinement.

When selecting a training loss function, L_2 loss struggles with symmetric objects, where multiple valid poses can produce identical visual appearances from the camera's perspective. Since L_2 loss enforces a single ground-truth pose, it incorrectly penalizes valid alternative poses, leading to inconsistent gradients and unstable training. ADD-S loss is a more suitable alternative as it accounts for symmetry and ensures robust training.

Choosing the right output representation also plays a crucial role in model performance. A disentangled representation of translation and rotation simplifies the learning process, making optimization more stable. Furthermore, some methods rely on real images for training, while others achieve strong performance using only synthetic data. The latter approach is more practical in real-world applications, where capturing real training data of the target object is difficult.

Among these methods, DeepIM and DenseFusion adopt an iterative refinement strategy, where the pose refinement network is applied multiple times to progressively refine the estimated pose. This approach improves pose accuracy, but at the cost of longer inference times, making it less suitable for real-time applications.

4.3. Filtering-based approach

The filtering-based approach conceptualizes 6D object pose tracking as a state estimation problem within the Bayesian filtering framework.

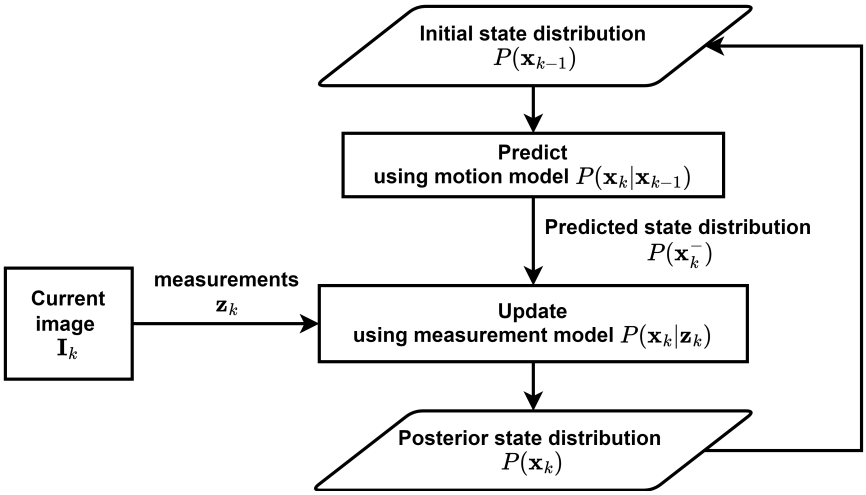


Figure 6: Overall schematic representation of filtering-based tracking approach.

Figure 6 provides a schematic overview of the filtering-based tracking approach. The state of the object is represented by a probability distribution, continuously updated as new sensor measurements become available. The Bayesian filtering involves two key steps: predict and update, which are performed recursively to estimate the state of an object over time. At each timestep k, pose estimation initiates from the initial state distribution $P(\mathbf{x}_{k-1})$, derived either from the preceding time step or defined by the user for the initial frame. The prediction step employs the motion model to estimate the object's next state based on its current state, yielding a predicted state distribution $P(\mathbf{x}_k^-)$. This motion model $P(\mathbf{x}_{k+1}|\mathbf{x}_k)$ characterizes the probability of transiting from current state to the next state, accounting for potential movements the object can make. In the update step, the predicted state distribution $P(\mathbf{x}_k^-)$ is further updated

using new measurements $\mathbf{z}_k$ from sensors, resulting in a posterior state distribution $P(\mathbf{x}_k)$. The measurement model $P(\mathbf{z}_k|\mathbf{x}_k)$ describes the probability of measuring $\mathbf{z}_k$ given state $\mathbf{x}_k$, allowing for adjustments based on actual sensor measurements.

There are three critical design choices in applying Bayesian filtering: the state vector, the motion model (process model), and the measurement model (observation model). In 6DoF object pose tracking, the state vector typically includes the object's position, orientation, and possibly derivatives like velocities and angular velocities. Common motion models include the Random Walk Model and the Constant Velocity Model. The Random Walk Model, favored for its simplicity, assumes that state changes are random and follow a Gaussian distribution, making it suitable when there is limited prior knowledge about the object's dynamics. The Constant Velocity Model assumes constant velocity in both translation and rotation, and is used when the object's motion is smooth and predictable. The measurement model varies depending on the type of measurements and is the key factor that differentiates various methods. Traditionally, edge-based measurements were used, but with the advent of depth cameras, many newer approaches now utilize depth values of object pixels.

Different approaches for performing Bayesian inference depend on the form of the motion and measurement models. For linear motion and measurement models, both subject to Gaussian noise, the inference problem can be optimally solved using a Kalman Filter. For nonlinear systems or when sensor measurements are nonlinear functions of the state, extensions of the Kalman Filter, such as the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF), are employed. The EKF linearizes the system dynamics and measurement functions, facilitating the application of the Kalman Filter framework to nonlinear systems. In contrast, the UKF approximates the probability distribution by propagating a set of sample points through nonlinear functions, offering a more accurate estimation of the object's state in highly nonlinear scenarios.

In cases where the probability distribution of the object's state is highly non-Gaussian or multimodal, non-parametric methods such as particle filters (PF) are used. Particle filters estimate the state using a set of weighted particles, $\{\mathbf{x}_k^{(i)}, w_k^{(i)}\}_{i=1}^N$, where each particle $\mathbf{x}_k^{(i)}$ represents a possible hypothesis of the object's state and $w_k^{(i)}$ is the associated weight. These particles are propagated through the system dynamics $P(\mathbf{x}_k|\mathbf{x}_{k-1})$, and their weights are updated based on the likelihood of the observation given the particle state $P(\mathbf{z}_k|\mathbf{x}_k^{(i)})$. The weights are then normalized to form a probability distribution, and resampling is performed to focus on the more likely particles, allowing the filter to effectively capture complex state distributions.

Among the filtering methods, PF is more frequently used for pose tracking from cameras. PF is well-suited for this application due to the non-Gaussian nature of the measurement model for 6DoF pose tracking, caused by sensor noise, potential occlusions, and object symmetries. However, the limitation of PF methods lies in the large number of particles required to represent the state distribution and the resulting high computational cost.

Table 7: Characteristics of the filtering-based instance-level 6D object pose tracking methods.

Method	Year	Input	Filter	State vector	Measurements	Motion model	GPU used?	Real-time capable?
Pupilli and Calway [132]	2006	RGB	PF	6D pose	Edges	Max distance Model	✗	✓
Klein and Murray [133]	2006	RGB	PF	6d pose	Edges	Random walk Model	✓	✓
Yoon [134]	2008	RGB	EKF	6D pose	Edges	Second-order autoregressive model	✗	Not reported
Teulière [135]	2010	RGB	PF	6D pose	Edges	Random walk Model	✗	Not reported
PCL [136]	2012	Depth	PF	6D pose	Positon and color of depth pixel	Not reported	✗	✗
Choi and Christensen [137]	2012	RGB	PF	6D pose	Edges	First-order autoregressive model	✗	✗
Choi and Christensen [59]	2013	RGB-D	PF	6D pose	Position, color, and normal of depth pixel	Not reported	✓	✓
Wüthrich [138]	2013	Depth	PF	6D pose, occlusion status	Depth value	Random walk Model	✗	✓
Isaac [139]	2016	Depth	RGF	6d pose, linear and angular velocity	Depth value	Constant velocity model	✗	✓

Table 7 provides an overview of filtering-based methods, highlighting their key characteristics such as state vector, motion model, measurements, and filter type. In the following sections, we review these methods in greater detail. Earlier filtering-based methods mainly used monocular cameras, limiting the visual features to edges. However, the recent introduction of depth sensors has made it possible to obtain real-time depth information, allowing the use of various geometric features, such as 3D point coordinates and surface normals. We begin by examining methods that use edges detected from RGB images as measurements, followed by a discussion of methods that exploit measurements from depth or RGB-D images.

4.3.1. RGB-based

Pupilli and Calway [132] first show the possibility of applying a PF to 3D edge-based tracking in real time. The state consists of camera position and orientation represented in a normalized quaternion. Motion model is the Max Distance Model from [140], which places equal probability to positions and rotations up to a certain range. Their measurement model groups line segments from a predefined 3D model into structures called 3D junctions, where multiple line segments (branches) converge at a central point. These junctions are projected onto the 2D image based on the camera's estimated pose. The system then matches these projections against the edge map of the camera frame, quickly assessing alignments as inliers or outliers by checking the proximity and orientation of the junction endpoints to the detected edges. The weight for a particle is then determined based on how many junctions match the observed edge map: $P(\mathbf{z}_k|\mathbf{x}_k) \propto \exp\left(\sum_{n=1}^{J} f(j_n)\right)$ where $f(j_n)$ represents the fractional contribution from each junction, and J is the total number of junctions. The contribution of each junction is computed as the number of branches that match expectations out of the total branches. By focusing on structured junctions rather than individual points or edges, the system demonstrates robustness in environments with complex and cluttered backgrounds.

While Pupilli and Calway [132] successfully track simple 3D models, they do not implement hidden line removal, which is crucial for tracking more complex models where self-occlusions frequently occur. Klein and Murray [133] address this by developing a Particle-Filter-based edge tracker capable of real-time tracking of complex 3D objects with self-occlusions, leveraging the GPU. In their model, the particle weight is calculated as $w \propto \exp\left(k\frac{d}{v}\right)$, where v represents the number of pixels forming the visible edges of the model, d is the number of these edge pixels that match edges detected in the image, and k is a tuning parameter. By utilizing hardware acceleration for hidden line removal and pose likelihood evaluations directly on the GPU, the filtering process is significantly sped up, and data transfer between the GPU and CPU is reduced, allowing tracking to be performed at video rate.

Teulière et al. [135] employ a particle filtering (PF) approach similar to Klein and Murray [133]. The principal distinction lies in their integration of a multiple hypothesis tracker that generates several camera pose hypotheses from low-level data by addressing ambiguities in edge matching. By applying the multiple hypothesis tracker to a select subset of particles—specifically those whose likelihoods surpass a defined threshold—their method effectively guides the particles towards the local maxima of the likelihood distribution. This strategy of local optimization through a multiple hypothesis tracker notably reduces the number of particles required, thus decreasing computational demands.

Choi and Christensen [137] further incorporate several improvements into the edge-based PF approach. For fast particle convergence, they employ keypoint features and initialize particles to highly probable states based on pose estimates solved from the PnP problem. Instead of employing a random walk motion model, as in [133, 135], they apply a first-order autoregressive (AR) model to guide particles more effectively. The measurement likelihood is derived from both the edge matching residual, as in [135], and the number of valid sample points in the edge correspondences, as in [133]. To reduce the number of particles needed, the particles are locally optimized using iteratively reweighted least squares (IRLS) to minimize the residual error between projected model edges and extracted image edges.

Yoon et al. [134] use an Extended Kalman Filter (EKF) to recursively optimize the pose by minimizing the error between the projection of the model's straight-line features and the corresponding scene edges in the image space. Although this approach does not explicitly handle multiple hypotheses like the Particle Filter (PF), it incorporates a backtracking scheme to prevent false model-to-scene correspondences from being used to update the predicted pose, thereby enhancing robustness in dynamic and complex environments. The backtracking strategy aims to find the largest subset of candidate features that meet the matching consensus criterion, ensuring that the maximum number of correct correspondences are retained while removing the inconsistent pairs.

4.3.2. Depth-based and RGB-D-based

Ueda [136] was the first to employ a particle filter (PF) for tracking the 6DoF pose using an object point cloud as a reference model across a sequence of RGB-D images. The particle likelihood is evaluated by measuring the squared distances between each point in the reference model and its closest point in the observed point cloud, as well as the distances between their corresponding colors. The effectiveness of this method in general settings is uncertain, as only one video of the experimental results is available. Despite the parallelization of its computationally intensive processes on a CPU, the system still fails to achieve real-time performance while using a large number of particles.

Compared to PCL tracking [136], Choi and Christensen [59] consider multiple renderings of the object instead of using a single reference point cloud for likelihood evaluation. The object is rendered according to the states of the

first V particles, and their likelihoods are evaluated by measuring Euclidean, normal, and color distances between the points of the rendered and observed point clouds. Remaining particles then identify and utilize the closest rendering from these initial V renderings to determine their likelihoods. Rendering the object model from various particle states allows this method to adapt more effectively to variations in pose. The most computationally demanding part of the likelihood evaluation is parallelized on GPU to maintain real-time performance.

Wüthrich et al. [138] present a method for tracking the 6D pose of objects manipulated by humans or robots, where occlusions are common. Their motion model adapts based on whether the manipulator provides control inputs. They employ a range camera for measurements and explicitly model occlusions with binary variables $\mathbf{o}_t^i$, indicating object visibility at each pixel i at time t. The measurement model, which predicts the depth at each pixel based on the object's pose and occlusion status, introduces strong nonlinearities. To handle this complexity, they employ a Rao-Blackwellised particle filter [141], which effectively marginalizes out the occlusion variables analytically when possible and samples only over potential poses. This approach not only reduces computational complexity but also enhances the accuracy of tracking the pose and occlusion states, allowing for real-time tracking on a single CPU core.

Isaac et al. [139] propose a 6D pose tracking method using depth images, focusing on overcoming the limitations of standard Gaussian filters (GFs) in handling fat-tailed measurement noise and high-dimensional measurements. The observation model is a weighted sum of a Gaussian model and a uniform model, accounting for unmodeled effects such as occlusions and outliers. A robustification method [142] is applied to enable the GF to work with this observation model. Additionally, they implement an approximation to the update step, which reduces computational complexity and allows for parallelization. The resulting robust Gaussian Filter (RGF) outperforms standard GFs and provides smoother and more accurate estimates compared to PF-based methods [138], especially in scenarios involving occlusions.

4.4. Hybrid approach

The end-to-end learning-based methods have shown great potential for robust and accurate 6DoF pose tracking in challenging scenarios. However, these methods are limited by their dependency on finite training samples, which can lead to inaccuracies. Additionally, they typically require powerful GPUs for training and inference, making real-time performance on ordinary devices, such as mobile phones, difficult to achieve. To address these limitations, some researchers have proposed hybrid methods that combine learning-based techniques with optimization or filter-based methods.

In hybrid methods that combine learners and optimizers, Zhong et al. [65] use learning-based methods to provide initial pose estimates and then refine them using optimization techniques. Other approaches replace certain modules in optimization-based methods with learning-based methods, using robustly learned features instead of handcrafted ones to enhance robustness in complex environments [143, 144]. For combinations of learners and filters, some methods incorporate neural network outputs as measurements within a Kalman Filtering framework [46, 47, 145]. Other works integrate learners into a Particle Filter framework, using learners to achieve accurate particle likelihood evaluation in complex scenes [57, 45].

Table 8 provides an overview of these hybrid methods. Detailed explanations of each method are provided below.

4.4.1. Combination of learners and optimizers

Tan et al. [65] sequentially combine a learner and an optimizer to leverage the complementary strengths of learning-based and optimization-based approaches. The learner uses random forests to predict the transformation parameters between consecutive frames, providing robust initial pose estimates that can handle large frame-to-frame movements and challenging conditions. The optimizer then refines these estimates by minimizing an energy function based on the displacement between 3D points on the object model and the corresponding points in the scene. This combination leverages the learner's robustness to avoid tracking failures and employs the optimizer to achieve better accuracy.

Zhong et al. [143] present a 6D object pose tracking method designed to handle heavy occlusions in monocular video sequences. They integrate a learning-based video object segmentation module with an optimization-based pose estimation module. The learner predicts the object mask in the current frame using information from the previous frame and the object's geometric model. The optimizer then converts these masks to foreground and background probability maps and refines the pose estimates by minimizing the energy function as proposed in PWP3D [36]. The

Table 8: Overview of hybrid instance-level 6D object pose tracking methods.

Method	Year	Input	Combination	Key characteristics
Krull [57]	2015	RGB-D	Learner+filter	Use Random Forest to learn 3D object coordinates for particle likelihood evaluation and proposal construction in PF framework
Tan [65]	2017	RGB-D	Learner+optimizer	Use Random Forest to learn initial pose estimates to be refined by optimizer
PoseRBPF [45]	2019	RGB/RGB-D	Learner+filter	Use auto-encoder network to learn a codebook for each discretized rotation, to aid particle likelihood evaluation in PF framework
Zhong [143]	2020	RGB	Learner+optimizer	Use video propagation network to predict soft foreground probability for the segmentation in region-based optimization
MaskUKF [46]	2021	RGB-D	Learner+filter	Use instance segmentation network output to extract point clouds, which serve as measurements of UKF
ROFT [47]	2022	RGB-D	Learner+filter	Use real-time optical flow to sync delayed outputs from low frame-rate networks for instance segmentation and 6D pose estimation with the RGB-D input stream, and incorporate them into linear KF and UKF for 6D object poses and velocities tracking.
Wu [145]	2022	RGB-D	Learner+filter	Use EKF predicted prior pose to render the input image for the Relative Pose Estimation Network
DeepAC [144]	2023	RGB	Learner+optimizer	Use a light weight network to predict boundary probability map for region-based optimization

refined pose from the optimizer is used to render a model-constrained mask for the next frame, creating a closed loop where the learner and optimizer mutually enhance each other's outputs. By using a robust video object segmentation network instead of simple color histograms for segmentation, their method achieves significantly better performance compared to previous works in handling heavy occlusions.

Wang et al. [144] present Deep Active Contours (DeepAC), a method that integrates the strengths of optimization-based techniques with learning-based approaches. The optimizer is built upon the correspondence line model from Stoiber et al. [146]. Each correspondence line samples points along the normal direction of the contour, and a lightweight neural network is used to predict a boundary probability map for these points. This map indicates the likelihood of each point being the true boundary. Pose parameters are iteratively refined by minimizing the energy function, defined as the negative log-likelihood of these probabilities. The neural network in the DeepAC model provides precise boundary localization and facilitates the iterative refinement of pose estimates, thereby significantly enhancing pose tracking accuracy and robustness in complex real-world conditions. The lightweight nature of the neural network ensures fast processing on mobile devices.

4.4.2. Combination of learners and filters

Inspired by Brachmann et al. [51], Krull et al. [57] train a random forest to regress 3D object coordinates from RGB-D images and incorporate these predictions into the PF tracking framework. The likelihood of each particle is evaluated based on how well its predicted pose aligns with both the image data and the predictions from the random forest. To decrease the number of particles needed to accurately represent the 6D posterior distribution, this method utilizes a proposal distribution rather than the original motion model during the sampling step. Informed by the output from the random forest and incorporating observed depth from the original images, previous particles, and the motion model, this proposal distribution strategically directs the particle sampling process, focusing computational resources on the most likely object poses. By applying the concept of 3D object coordinate regression to 6D pose tracking and leveraging predicted object coordinates within a proposal distribution, this method effectively handles fast movements, occlusions, and lighting variations.

Deng et al. [45] propose PoseRBPF, an efficient and robust 6D pose tracking method leveraging a Rao-Blackwellized particle filter alongside an auto-encoder network. PoseRBPF efficiently decouples the estimation of 3D rotation and 3D translation of objects. Each particle samples 3D translation and estimates the distribution over 3D rotations conditioned on the image bounding box corresponding to the sampled translation. To capture subtle changes in orientation, the rotation space is discretized in a fine-grained manner, and an auto-encoder network is trained to construct a codebook of feature embeddings for the discretized rotations. During tracking, the estimated translation for each particle is used to adjust the image's bounding box, which is then passed through the auto-encoder network to generate an embedding. PoseRBPF compares each bounding box embedding against the learned codebook to update distributions over time. PoseRBPF estimates not only pose but also its uncertainty, leading to an accurate tracker for objects with

diverse symmetries and under varying conditions of occlusion and lighting.

Learning-based techniques have been integrated into a Particle Filter framework to aid particle likelihood evaluation, as demonstrated in [57, 45]. Another category of methods [46, 47, 145] use neural network output to generate measurements or directly as measurements in variants of a Kalman Filter. For instance, MaskUKF [46] combines deep learning-based object segmentation with an Unscented Kalman Filter (UKF) to enable real-time tracking of the 6D pose and velocity of objects. The binary masks generated by the segmentation network are used to extract point clouds from RGB-D images, which then serve as measurements to the UKF for tracking. Compared to the deep-learning-based methods in section 4.2.2, this approach does not require extensive 6D pose annotations or specific training for each object. A segmentation network trained once for all the objects of interest is enough to perform tracking.

Piga et al. [47] propose Real-Time Optical Flow-Aided 6D Object Pose and Velocity Tracking (ROFT). ROFT uses real-time optical flow to synchronize the delayed outputs from low frame rate Convolutional Neural Networks, which are used for instance segmentation and 6D object pose estimation, with the RGB-D input stream, enabling real-time 6D object pose and velocity tracking. The system first uses a linear KF to estimate the object's velocity from the synchronized segmentation masks and optical flow data. These velocity estimates are then integrated with the pose measurements from the pose estimation network in the UKF to update and track the object's pose over time. The velocities estimated from optical flow provide the UKF with valuable 6D motion direction information, which is especially useful for tracking fast moving objects.

Wu et al. [145] address fast motion scenarios by integrating a Relative Pose Estimation Network from an end-to-end 6D pose tracker with an Extended Kalman Filter (EKF) to improve tracking accuracy. At each frame, the EKF predicts the state using the motion model, generating a prior 6D pose estimate. This predicted pose is used to render an image of the object, which, along with the observed frame, is fed into the relative pose estimation network to compute the relative pose transformation. This transformation updates the pose estimate and serves as a measurement for the EKF update step. Rendering the relative pose estimation network's input with the EKF predicted state improves tracking performance in fast motion, as it provides a pose closer to the actual object pose, enhancing the network's prediction accuracy.

5. Performance analysis

A comprehensive comparison of performance across methods is essential for understanding the current state of instance-level 6D object pose tracking and for identifying the most suitable approaches for different applications. In this section, we evaluate representative tracking methods on the datasets introduced in Section 3, including Choi [59], YCB-Video [12], OPT [61], Laval 6DOF [42], RBOT [37], BCOT [62], YCBInEOAT [43], and InOT6D [63].

We focus on the accuracy evaluation using the metrics defined in Section 2.2. The quantitative results presented here are mainly sourced from the respective research papers. Where available, we also include runtime performance data. Despite variations in experimental settings and hardware platforms, these results provide a useful indication of the efficiency of different methods.

After reviewing results on each dataset individually, we provide an overview table, Table 17, that aggregates the accuracy and runtime of all evaluated methods, followed by a concluding subsection that summarizes key findings and trends.

5.1. Choi

The synthetic dataset of Choi [59] is the most commonly used and well investigated dataset by earlier 6DoF pose tracking methods. We present the Root Mean Square (RMS) error in the X, Y, and Z directions and in the Roll, Pitch, and Yaw angles in Table 9. The average runtime per frame is also presented if it is reported in the respective work. All the trackers are initialized with the ground truth, and all the results are adopted from the respective publications.

Table 9 shows that the region-based optimization method ICG [67] performs best on the Choi dataset among all of the displayed methods, achieving RMS error of 0.04 mm for translation and 0.04 degree for rotation. The particle filter-based methods by Choi [59] and Krull [57] have a significantly higher runtime compared to other learning-based and optimization-based methods. The hybrid method proposed by Tan et al. in 2017 [65], which refines the pose estimate from a random forest-based learner using an optimizer, performs much better than their earlier learning-based method Tan et al. in 2014 [64]. This comparison demonstrates the advantage of combining the strengths of a learner and an optimizer.

Table 9: RMS errors for translation (mm), rotation (degree) and the runtime (ms) of the tracking results on the Choi dataset [59].

Method		Choi [59]	Krull [57]	Tan [64]	Akkaladevi [58]	Tan [65]	Kehl [66]	ICG [67]
(a)*Kinect Box*	*X*	1.84	0.83	1.54	0.30	0.15	0.76	**0.05**
	Y	2.23	1.67	1.90	0.49	0.19	1.09	**0.11**
	Z	1.36	0.79	0.34	0.31	0.09	0.38	**0.03**
	Roll	6.41	1.11	0.42	0.21	0.09	0.17	**0.02**
	Pitch	0.76	0.55	0.22	0.27	0.06	0.18	**0.02**
	Yaw	6.32	1.04	0.68	0.23	0.04	0.20	**0.02**
	Time	166	143	1.5	1.71	2.2	8.10	-
(b)*Milk*	*X*	0.93	0.51	1.23	0.63	0.15	0.76	**0.05**
	Y	1.94	1.27	0.74	1.19	0.19	1.09	**0.11**
	Z	1.09	0.62	0.24	0.48	0.09	0.38	**0.03**
	Roll	3.83	2.19	0.50	0.19	0.09	0.17	**0.02**
	Pitch	1.41	1.44	0.28	0.28	0.06	0.18	**0.02**
	Yaw	3.26	1.90	0.46	0.27	0.04	0.20	**0.02**
	Time	2205	134	135	1.5	2.1	8.54	-
(c)*Orange Juice*	*X*	0.96	0.52	1.10	0.39	0.11	0.50	**0.04**
	Y	1.44	0.74	0.94	0.37	0.09	0.69	**0.03**
	Z	1.17	0.63	0.18	0.37	0.09	0.17	**0.02**
	Roll	1.32	1.28	0.35	0.12	0.08	0.12	**0.05**
	Pitch	0.75	1.08	0.24	0.17	0.08	0.20	**0.03**
	Yaw	1.39	1.20	0.37	0.15	0.08	0.19	**0.06**
	Time	117	129	1.5	1.69	2.2	8.79	-
(d)*Tide*	*X*	0.83	0.69	0.73	0.42	0.08	0.34	**0.02**
	Y	1.37	0.81	0.56	0.51	0.09	0.49	**0.03**
	Z	1.20	0.81	0.24	0.64	0.07	0.18	**0.01**
	Roll	1.78	2.10	0.31	0.22	0.05	0.15	**0.03**
	Pitch	1.09	1.38	0.25	0.29	0.12	0.39	**0.04**
	Yaw	1.13	1.27	0.34	0.30	0.05	0.37	**0.03**
	Time	111	116	1.5	1.7	2.2	9.42	-
Mean	Transl.	1.36	0.82	0.81	**0.50**	0.10	0.51	**0.04**
	Rot.	2.45	1.38	0.37	**0.22**	0.07	0.26	**0.04**
	Time	132	131	**1.5**	1.7	2.2	8.71	-

Table 10: The quantitative evaluation for 6D pose tracking methods on the YCB-Video dataset [12] by using AUC(ADD_0.1m) and AUC(ADD-S_0.1m). Better results are indicated in bold type. Objects with rotational symmetry are shown in bold.

Method	Wüthrich [138]		RGF [142]		PoseRBPF [45]		DeepIM [119]		*se*(3)-TrackNet [43]		HFF6D [124]		FCR-TrackNet [125]		ICG [67]		ICG+ [106]		FoundationPose [23]	
Modality	Depth		Depth		RGBD		RGB		RGBD		RGBD		RGBD		RGBD		RGBD		RGBD	
Initial pose from	Groundtruth		Groundtruth		PoseCNN		Groundtruth		Groundtruth		PoseCNN		Groundtruth		Groundtruth		Groundtruth		Groundtruth	
Re-initialization (Total)	No		No		Yes (2)		Yes (290)		No		No		No		No		No		No	
Objects	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S
002_master_chef_can	55.6	90.7	46.2	90.2	90.5	95.1	89.0	93.8	93.9	96.3	91.8	95.8	**95.9**	97.5	66.4	89.7	94.7	**98.0**	93.6	97.0
003_cracker_box	96.4	97.2	57.0	72.3	88.2	93.0	88.5	93.0	96.5	97.2	96.5	97.2	96.4	97.2	82.4	92.1	96.8	**98.4**	**96.9**	97.8
004_sugar_box	97.1	97.9	50.4	72.7	92.9	95.5	94.3	96.3	**97.6**	98.1	**97.6**	98.2	97.0	97.8	96.1	98.4	96.6	**98.5**	96.9	98.2
005_tomato_soup_can	64.7	89.5	72.4	91.6	90.0	93.8	89.1	93.2	95.0	97.2	94.2	97.3	**98.0**	**98.3**	73.2	97.3	95.2	98.1	96.3	98.1
006_mustard_bottle	**97.1**	98.0	87.7	98.2	91.9	96.3	92.0	95.1	95.8	97.4	96.9	97.8	97.0	97.8	96.2	98.4	97.1	**98.7**	**97.3**	98.4
007_tuna_fish_can	69.1	93.3	28.7	52.9	91.1	95.3	92.0	96.4	86.5	91.1	85.3	91.5	88.0	93.2	73.2	95.8	94.1	97.1	**96.9**	**98.5**
008_pudding_box	96.8	97.9	12.7	18.0	85.8	92.0	80.1	88.3	97.9	98.4	98.0	98.4	97.9	98.4	73.8	88.9	80.4	90.5	**97.8**	**98.5**
009_gelatin_box	97.5	98.4	49.1	70.7	96.3	97.5	92.0	94.4	97.8	98.4	**98.1**	98.6	98.0	98.5	97.2	**98.8**	96.8	98.1	97.7	98.5
010_potted_meat_can	83.7	86.7	44.1	45.6	68.7	77.9	78.0	88.9	77.8	84.2	50.8	76.8	**96.1**	97.6	93.3	97.3	95.4	**98.1**	95.1	97.7
011_banana	86.3	96.1	93.3	97.7	74.2	86.9	81.0	90.5	94.9	97.2	89.7	96.4	95.8	97.7	95.6	**98.4**	92.8	98.2	**96.4**	**98.4**
019_pitcher_base	97.3	97.7	**97.9**	98.2	86.8	94.2	90.4	94.7	96.8	97.5	96.7	97.4	97.0	97.6	97.0	98.8	97.2	**98.9**	96.7	98.0
021_bleach_cleanser	95.2	97.2	95.9	97.3	86.0	93.0	81.7	90.5	**95.9**	97.3	93.5	96.7	94.4	97.6	92.6	97.5	91.6	97.0	95.5	**97.8**
024_bowl	30.4	82.4	24.2	82.4	25.5	94.2	38.8	90.6	80.9	94.5	-	95.6	87.0	97.1	74.4	**98.4**	84.1	97.9	**95.2**	97.6
025_mug	83.2	93.3	60.0	71.2	90.9	97.1	83.2	92.0	91.5	96.9	90.73	96.4	95.0	97.7	**95.6**	**98.5**	**95.6**	98.4	**95.6**	97.9
035_power_drill	97.1	97.8	**97.9**	98.3	93.9	96.1	85.4	92.3	96.4	97.4	96.53	97.5	97.1	97.8	96.7	98.5	96.8	**98.7**	96.9	98.2
036_wood_block	95.5	96.9	45.7	62.5	20.1	89.1	44.3	75.4	95.2	96.7	-	96.7	93.2	95.8	93.5	97.2	91.7	96.7	93.2	97.0
037_scissors	4.2	16.2	20.9	38.6	76.1	85.6	70.3	84.5	95.7	97.6	93.86	96.5	**95.8**	**97.6**	93.5	97.3	94.9	**97.6**	94.8	97.5
040_large_marker	35.6	53.0	12.2	18.9	92.0	97.1	80.4	91.2	92.2	96.0	90.08	96.9	87.9	93.6	88.5	97.8	94.0	98.1	**96.9**	**98.6**
051_large_clamp	61.2	72.3	62.8	80.1	48.5	94.8	73.9	84.1	**94.7**	96.9	-	96.4	93.8	96.5	91.8	96.9	93.8	**97.8**	93.6	97.3
052_extra_large_clamp	93.7	96.6	67.5	69.7	40.3	90.1	49.3	90.3	91.7	95.8	-	91.3	90.8	95.1	85.9	94.3	91.8	97.0	**94.4**	**97.5**
061_foam_brick	96.8	98.1	70.0	86.5	81.1	95.7	91.6	95.5	93.7	96.7	-	97.9	97.4	97.9	96.2	98.5	94.0	97.9	**97.9**	**98.6**
ALL	78.0	90.2	59.2	74.3	80.8	93.3	79.3	91.0	93.0	95.7	-	95.6	94.5	96.7	86.4	96.5	94.3	**97.9**	**96.0**	**97.9**
Speed (fps)	12.9		11.8		5.0		12.0		90.9		306.2		89.6		**788.4**		111.7		32	

5.2. YCB-Video

Table 10 presents the evaluation of state-of-the-art 6D pose tracking methods using the YCB-Video dataset [12]. The *ADD* and *ADD-S* metrics of ten methods from four categories are compared: filtering-based [138, 142], hybrid [45], deep-learning-based [119, 43, 124, 125, 23], and optimization-based [67, 106]. The results of Wüthrich [138] and RGF [142] are from the work of Wen et al. [43], where they evaluated Wüthrich [138] and RGF [142] using publicly available source code, while the other results are from their respective publications.

Table 11: The quantitative evaluation for 6D pose tracking methods on the OPT dataset [61] by using AUC(ADD_0.2d) without any re-initialization. Better results are indicated in bold type. Objects with rotational symmetry are shown in bold.

Method	**Soda**	Chest	Ironman	House	Bike	Jet	Avg.
PWP3D [36]	5.87	5.55	3.92	3.58	5.36	5.81	5.01
Bugaev [91]	14.85	14.97	14.71	14.48	12.55	17.17	14.79
Tjaden [37]	8.86	11.76	11.99	10.15	11.90	13.22	11.31
Zhong [104]	9.01	12.24	11.21	13.61	12.83	15.44	12.39
Li [95]	9.00	14.92	13.44	13.60	12.85	10.64	12.41
LDT3D [97]	4.2	9.2	3.26	4.05	7.63	8.65	6.16
SRT3D [38]	15.64	16.30	17.41	16.36	13.02	15.64	15.39
ICG [67]	15.32	15.85	17.86	17.92	16.36	15.90	16.54
ICG+ [106]	**16.72**	17.37	**17.93**	**18.64**	**16.92**	**17.82**	**17.57**
DeepAC [144]	15.71	**17.63**	17.58	18.01	13.91	17.17	16.67
Chen [107]	16.52	17.03	17.75	16.71	13.04	17.63	16.45

Table 10 shows that FoundationPose [23] achieves the highest ADD-S metric, while both FoundationPose and ICG+ [106] perform best on the ADD metric. A key advantage of FoundationPose is its ability to be applied directly at test time to a novel object without fine-tuning, requiring only its CAD model. In contrast, other instance-level deep learning-based methods rely on CAD models for generating training data. Despite this reduced assumption, FoundationPose still achieves state-of-the-art performance, demonstrating strong generalization ability. This robustness is driven by its large-scale synthetic training and transformer-based architecture, which enable effective feature extraction and pose refinement across a wide range of objects.

Methods that do not leverage texture information struggle with the ADD score, particularly for rotationally symmetric objects, where geometry alone is not conclusive for determining rotation. By utilizing texture information with keypoints, ICG+ [106] significantly improves its ADD score compared to the original ICG approach [67] without multi-region and texture.

Table 10 also reveals that deep learning-based and optimization-based methods generally have better accuracy than filtering-based methods, and RGB-D based methods outperform those based solely on RGB or depth. In addition to tracking quality, Table 10 also compares the average frame rates. Filtering-based methods have much lower frame rates compared to deep learning-based and optimization-based methods.

5.3. OPT

Table 11 shows the experimental results on the OPT dataset [61]. The evaluated 6D pose tracking methods are primarily optimization-based and utilize various sources of information such as region, edge, keypoint, and depth. The hybrid method DeepAC, which combines learning and optimization, is also included. For the evaluation, the area under curve of the ADD metric is used with a threshold of $e_t = 0.2d$, where d is the largest distance between model vertices. The final values are scaled between 0 and 20 and are referred to as AUC scores.

Table 11 shows that ICG+ has the highest average AUC score of 17.82. ICG+ also demonstrates consistent high performance across most objects.

5.4. Laval 6DOF

Table 12 evaluates the tracking accuracy of the random-forest-based learning method by Tan et al. [118] and two deep-learning-based methods, D6DT[41] and Garon [42], on the Laval dataset [42]. The accuracy metric, as defined in section 2.2.2, includes translational and rotational errors.

Table 12: Translational and rotational errors of 6D pose tracking methods on Laval dataset [42].

Stability scenario	**Translation (mm/frame)**			**Rotation (degree/frame)**		
	Near	Far	Occluded	Near	Far	Occluded
Garon [42]	**0.56**	**0.68**	**0.72**	**0.52**	**0.59**	**0.76**
D6DT [41]	0.93	1.06	1.24	1.13	1.23	1.49
Tan [118]	1.20	1.31	1.53	1.30	1.44	1.92

Occlusion scenario	**Translation (mm)**						**Rotation (degrees)**					
Occlusion %	0	15	30	45	60	75	0	15	30	45	60	75
Garon [42]	**7.4**	9.8	**11.5**	**12.5**	**15.5**	**24.2**	**3.6**	**5.9**	**7.9**	**10.0**	**12.6**	**22.2**
D6DT [41]	7.4	11.2	18.9	26.8	38.1	55.0	5.3	8.8	17.7	28.2	41.7	49.8
Tan [118]	8.2	**8.5**	15.9	138	186	213	4.0	7.4	33.1	70.3	89.5	88.0

Interaction scenario	**Translation (mm)**				**Rotation (degrees)**			
Speed (per frame)	(0, 12.5]	(12.5, 25]	(25, 37.5]	(37.5, 50]	(0, 19]	(19, 37]	(37, 56]	(56, 75]
Garon [42]	8.2	10.3	11.1	13.4	**3.7**	**5.8**	**3.6**	**5.8**
D6DT [41]	9.5	10.2	**10.3**	**12.4**	7.8	9.5	13.4	11.8
Tan [118]	**8.1**	**8.5**	10.7	67.1	4.5	6.0	8.1	10.1

The trackers are assessed in three scenarios: stability, occlusion, and interaction. Except for the "free-hard" scenario, the tracker is re-initialized at the ground truth pose every 15 frames. In the "free-hard" sequence, the tracker resets to the ground truth pose upon detecting a tracking failure. For the interaction scenario, given the varying object speeds, the translational and rotational inter-frame displacements are computed, and performance metrics are reported as a function of these displacements.

As shown in Table 12, the Garon method demonstrates superior robustness and accuracy compared to D6DT and Tan. Garon shows the smallest errors in both translation and rotation across most scenarios and conditions. While D6DT generally performs well, it exhibits higher errors, particularly in high occlusion conditions. Tan tends to have the highest errors across most conditions.

5.5. RBOT

Table 13: Tracking success rate on the RBOT dataset [37]. Better results are indicated in bold type.

Method	Regular	Dynamic Light	Noisy	Unmodeled Occlusion	Avg.	Runtime(ms)
Tjaden [37]	79.9	81.2	56.6	73.3	72.8	15.5~21.8
Zhong [104]	82.7	81.3	63.6	78.4	76.5	41.2
Li [95]	85.8	86.7	71.4	80.3	81.1	32.1
Huang [93]	86.9	87.3	65.0	83.6	80.7	33.1
Sun [94]	88.1	88.8	80.5	85.1	85.6	40~50
Huang [105]	89.9	90.7	69.6	88.9	84.8	33.1
SRT3D [38]	94.2	94.6	81.7	93.2	90.9	**1.1**
LDT3D [97]	95.2	95.4	83.2	**94.9**	92.2	13.6
ICG [67]	94.4	94.9	82.4	93.7	91.4	-
DeepAC [144]	**95.6**	**95.6**	**88.0**	94.0	93.3	133.7
FSCGE [63]	94.7	95.0	86.0	94.1	92.5	14.8
Chen [107]	-	-	-	-	**95.4**	6.6

Table 13 compares the accuracy of different methods on the RBOT dataset across two categories: 1) optimization-based approaches, including region-based [37, 104, 38, 67, 63, 107] and edge-based [95, 93, 94, 105, 97] methods, and 2) the latest hybrid method, DeepAC [144], which combines optimization with learning. The accuracy is computed using the 5cm 5° metric, following the standard tracking evaluation protocol [37]. A frame is considered successfully tracked if the translation and rotation errors are less than 5 cm and 5°, respectively. Otherwise, it is considered a tracking failure, and the algorithm is re-initialized with the ground-truth pose.

Chen et al. [107] achieve the highest average success rate of 95.4%, while DeepAC [144] ranks second with 93.3%. In terms of efficiency, the contrast is notable: Chen et al.'s method runs at 6.6 ms per frame, making it well suited for real-time applications, whereas DeepAC requires 133.7 ms per frame, which limits its practicality in time-critical scenarios despite its accuracy and robustness. For comparison, SRT3D [38] achieves a runtime of only 1.1 ms

per frame while still maintaining a strong average success rate of 90.9%, highlighting a favorable trade-off between speed and accuracy.

5.6. BCOT

Table 14: Comparison of 6D pose tracking methods on the BCOT benchmark [62]. The metrics include the ADD scores with 2%, 5%, 10% of the object diameter, cm-degree scores, and the runtime per frame.

Method	ADD_			cm-degree		Runtime (ms)
	0.02d	0.05d	0.1d	5-5	2-2	
Wang [92]	5.5	32.7	64.6	54.4	12.4	8.8
Tjaden [37]	11.7	31.6	57.1	77.1	40.8	34.6
Huang [93]	12.0	31.3	57.5	84.1	45.1	33.0
Li [96]	9.1	31.5	58.1	95.0	38.5	18.9
Huang [105]	15.6	39.8	66.1	87.1	51.4	34.8
Li [95]	14.4	38.1	65.7	87.0	50.2	38.5
RBGT [146]	10.9	45.5	76.9	89.0	46.0	**3.5**
SRT3D [38]	12.5	49.4	82.1	93.1	53.6	-
LDT3D [97]	15.3	52.1	82.7	93.8	63.2	-
DeepAC [144]	**24.4**	**66.2**	**92.3**	94.0	**65.5**	-
Chen [107]	14.1	51.3	82.9	**95.4**	63.4	-

Table 14 compares the performance of various 6D pose tracking methods on the BCOT benchmark [62]. The metrics include ADD scores with different thresholds, cm-degree scores, and the runtime per frame in milliseconds. In addition to the commonly used, relatively relaxed metrics ADD_0.1d and 5cm 5°, more rigorous criteria such as ADD_0.02d, ADD_0.05d, and 2cm 2° scores are included to evaluate high-precision tracking capabilities. For all the metrics in the table, re-initialization is performed when the rotation error is larger than 5° or the translation error is larger than 5cm. The accuracy metrics for SRT3D [38], LDT3D [97], and DeepAC [144] are cited from the DeepAC [144] paper, while other metrics are sourced from the BCOT [62] paper. The metrics for Chen et al. [107] are reported in its own paper.

The results show that DeepAC [144] achieves the best performance across all ADD metrics and the 2cm 2° score, with substantial margins over other optimization-based methods. In contrast, Chen et al.'s method [107] obtains the highest accuracy on the standard 5cm 5° metric (95.4%), slightly surpassing DeepAC (94.0%). These findings indicate that DeepAC excels in high-precision tracking, whereas Chen et al.'s method delivers state-of-the-art performance under the more commonly used, less strict evaluation threshold.

5.7. YCBInEOAT

The results in Table 15 compare the performance of 6D pose tracking methods on the YCBInEOAT [43] dataset in a robotic manipulation setting. Early filtering-based methods such as RGF [142] and Wüthrich [138] perform significantly worse, with average ADD scores below 80%. In contrast, more recent deep-learning-based approaches, including se(3)-TrackNet [43], FCR-TrackNet [125], HFF6D [124], and FoundationPose [23], achieve substantially higher accuracy. Among them, FCR-TrackNet reaches the best overall ADD score (93.2%), while FoundationPose achieves the best ADD-S score (96.4%). Notably, FoundationPose requires only the CAD model at test timeand is not trained on object-specific data, highlighting the strong generalization enabled by transformer-based architectures and large-scale pre-training.

5.8. InOT6D

Table 16 compares the accuracy of representative tracking methods on the InOT6D [63] dataset, which evaluates robustness in industrial scenarios with six variants: Regular (REG), Dynamic light (DYN), Motion blur (MOT), Unmodeled occlusion (UOC), Modeled occlusion (MOC), and Color ambiguity (COR). The accuracy is measured using three metrics: mean rotation error (R, °), mean translation error (t, mm), and success rate (SR, %). A frame is considered successfully tracked if both the rotation error is below 5° and the translation error is below 5 mm. Unlike

Table 15: Comparison of 6D pose tracking methods utilizing the AUC(ADD_0.1m) and AUC(ADD- S_0.1m) metrics on the YCBInEOAT dataset. Best results are indicated in bold. Objects with rotational symmetry are shown in bold.

Objects	RGF [142]		Wüthrich [138]		se(3)-TrackNet [43]		FCR-TrackNet [125]		HFF6D [124]		FoundationPose [23]	
	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S
003_cracker_box	34.8	55.4	79.0	88.1	90.8	94.1	**91.4**	94.6	90.38	93.86	91.32	**95.10**
004_sugar_box	15.8	16.9	86.8	92.8	92.4	94.8	**93.6**	95.9	92.35	95.59	91.45	**95.96**
005_tomato_soup_can	15.1	26.4	63.7	93.2	93.4	**97.0**	92.4	95.8	92.68	96.56	**94.14**	96.67
006_mustard_bottle	56.5	60.2	91.3	95.3	97.0	97.9	**97.1**	**98.1**	95.82	97.18	91.71	96.58
021_bleach_cleanser	29.4	45.0	61.5	69.0	89.6	94.4	92.1	95.7	93.19	95.82	**96.34**	**97.89**
ALL	30.0	40.0	78.3	89.2	92.6	95.5	**93.2**	96.0	92.88	95.80	93.09	**96.42**

Table 16: Performance on the InOT6D dataset under six conditions: Regular (REG), Dynamic light (DYN), Motion blur (MOT), Unmodeled occlusion (UOC), Modeled occlusion (MOC), and Color ambiguity (COR). Rotation error (R, °), translation error (t, mm), and success rate (SR, %) are reported. Best in bold.

Method	REG			DYN			MOT			UOC			MOC			COR			Avg.		
	R	t	SR	R	t	SR	R	t	SR	R	t	SR	R	t	SR	R	t	SR	R	t	SR
Tjaden [37]	–	–	84.9	–	–	83.8	–	–	78.5	–	–	71.3	–	–	–	–	–	35.8	–	–	70.9
Sun [94]	4.03	15.28	83.3	4.45	14.70	81.4	4.59	14.42	80.1	5.55	20.14	76.5	–	–	–	9.80	34.67	53.7	5.68	19.84	75.0
Huang [105]	2.56	8.74	87.6	2.78	8.35	86.3	3.38	11.01	83.0	3.39	11.63	82.4	–	–	–	5.78	17.39	67.1	3.58	11.42	81.3
SRT3D [38]	1.74	3.37	93.8	1.89	4.22	92.9	2.57	9.62	88.8	2.11	3.99	90.7	2.00	3.75	91.6	3.34	6.70	79.3	2.28	5.28	89.5
LDT3D [97]	1.39	2.21	94.1	1.55	2.81	93.3	2.58	7.66	88.2	1.98	3.64	90.1	–	–	–	3.39	7.40	81.1	2.18	4.74	89.4
ICG+(d-) [106]	1.79	3.77	94.3	2.03	4.64	92.6	2.69	10.00	88.5	2.77	5.89	86.7	2.06	4.25	92.6	3.09	7.03	83.4	2.41	5.93	89.7
DeepAC [144]	1.52	4.56	93.5	1.60	4.55	93.0	3.14	7.76	81.6	2.29	5.65	87.7	–	–	–	3.34	6.59	80.1	2.38	5.82	87.2
FSCGE [63]	**1.48**	**3.20**	**96.7**	**1.45**	**3.56**	**96.8**	**2.22**	**8.47**	**92.3**	**1.81**	**3.72**	**93.6**	**1.69**	**3.49**	**95.1**	**1.92**	**4.76**	**92.9**	**1.76**	**4.53**	**94.6**

RBOT, no re-initialization is performed once tracking is lost, making this evaluation stricter and closer to real-world industrial conditions.

The methods include two edge-based optimization methods Sun [94] and Huang [105], and five region-based optimization methods, Tjaden [37], SRT3D [38], LDT3D [97], ICG+(d-) [67], and FSCGE [63], and one hybrid method DeepAC [144]. Here, ICG+(d-) denotes a variant of the ICG tracker in which the depth modality is disabled.

The best performance is achieved by FSCGE, which obtains the lowest errors and the highest average SR of 94.6%. Its robustness across all variants, particularly MOT (92.3%) and occlusion (UOC = 93.6%, MOC = 95.1%), demonstrates the effectiveness of its multi-feature fusion framework. All other methods show noticeable degradation in the more challenging variants, particularly COR, MOT, and UOC. By adaptively combining statistical color cues with geometric edge correspondences based on observations, FSCGE mitigates the weaknesses of individual modalities and maintain stable performance where other methods fail. Compared to SRT3D, ICG+(d-) additionally leverages texture information and multi-region modality, however, ICG+(d-) performs very similar as its earlier work SRT3D, without the clear advantage it shows on textured benchmarks, since InOT6D primarily features textureless industrial objects where keypoint-based cues are less effective.

5.9. Summary

In the previous subsections, we examined the accuracy and runtime performance of instance-level 6D pose tracking methods across several widely used datasets individually. To facilitate clearer comparison, Table 17 provides an aggregated overview of 35 representative methods, organized by category, with their performance summarized across datasets alongside reported runtime:

1) Region-based methods remain highly competitive. ICG [67] and ICG+ [106] perform best on the Choi and OPT [61] datasets. The comparison between ICG+ [106] and ICG [67] demonstrates the advantage of combining multiple cues. ICG [67] exploits region color around the silhouette and geometry information. ICG+ [106] further incorporates surface texture information using keypoint feature, leading to better performance compared to ICG [67]. Chen [107] reaches state-of-the-art accuracy on RBOT and BCOT (approximately 95%) with an efficient runtime of about 6.6 ms. Furthermore, FSCGE [63] achieves the best results on InOT6D (94.6%), highlighting that multi-feature fusion is effective in handling industrial texture-less object tracking with multiple challenges such as unmodelled occlusion, motion blur and color similarity.

Table 17: Overall performance (dataset averages only) for all mentioned methods across datasets. Metrics follow each dataset's convention: Choi = RMS (translation mm / rotation deg); YCB-Video = AUC (ADD/ADD-S, 0.1m); OPT = AUC(ADD, 0.2d) scaled to [0,20]; Laval = error (translation / rotation); RBOT = SR (%) under 5° 5cm metric; BCOT = SR (%) under 5° 5cm metric; YCBInEOAT = AUC (ADD/ADD-S, 0.1m); InOT6D = SR (%) under 5° 5cm metric. Best results are shown in bold, while second best results are underscored. "–" indicates unavailable. RF: Random Forest; DL: Deep Learning; LF: Combination of learners and filters; LO: combination of learners and optimizers.

Category		Method	Year	Input	OPT	RBOT	BCOT	InOT6D	Choi		Laval		YCB-Video		YCBInEOAT		Runtime (ms)
									t	R	t	R	Add	ADD-S	Add	ADD-S	
Optimization-based	edge	Bugaev [91]	2018	RGB	14.79	-	-	-	-	-	-	-	-	-	-	-	-
		Wang [92]	2019	RGB	-	-	54.4	-	-	-	-	-	-	-	-	-	8.8
		Huang20 [93]	2020	RGB	-	80.7	84.1	-	-	-	-	-	-	-	-	-	33.0
		Sun [94]	2021	RGB	-	85.6	-	75	-	-	-	-	-	-	-	-	40 ~ 50
		Li [95]	2021	RGB	12.41	81.1	87	-	-	-	-	-	-	-	-	-	32.1
		Li [96]	2021	RGB	-	-	95	-	-	-	-	-	-	-	-	-	18.9
	region	PWP3D [36]	2012	RGB	5.01	-	-	-	-	-	-	-	-	-	-	-	-
		Kehl[66]	2017	RGB-D	-	-	-	-	0.51	0.26	-	-	-	-	-	-	8.71
		Tjaden [37]	2018	RGB	11.31	72.8	77.1	70.9	-	-	-	-	-	-	-	-	15.5 ~ 21.8
		Zhong [104]	2020	RGB	12.39	76.5	-	-	-	-	-	-	-	-	-	-	41.2
		RBGT [146]	2020	RGB	-	-	-89	-	-	-	-	-	-	-	-	-	3.5
		Huang21 [105]	2021	RGB	-	84.8	87.1	81.3	-	-	-	-	-	-	-	-	33.1
		LDT3D [97]	2022	RGB	6.16	92.2	93.5	89.4	-	-	-	-	-	-	-	-	13.6
		ICG [67]	2022	RGB-D	16.54	91.4	-	-	**0.04**	**0.04**	-	-	86.4	96.5	-	-	1.3
		SRT3D [38]	2022	RGB	15.39	90.9	93.1	89.5	-	-	-	-	-	-	-	-	**1.1**
		ICG+	2023	RGB-D	**17.57**	-	-	89.7(d-)	-	-	-	-	94.3	**97.9**	-	-	8.95
		FSCGE [63]	2024	RGB	-	92.5	-	**94.6**	-	-	-	-	-	-	-	-	14.8
		Chen [107]	2025	RGB	16.45	**95.4**	**95.4**	-	-	-	-	-	-	-	-	-	6.6
Learning-based	RF	Tan14 [64]	2014	Depth	-	-	-	-	0.81	0.37	-	-	-	-	-	-	1.5
		Tan15 [118]	2015	Depth	-	-	-	-	-	-	18.07	32.1	-	-	-	-	-
		Akkaladevi [58]	2016	Depth	-	-	-	-	0.50	0.22	-	-	-	-	-	-	1.7
	DL	D6DT [41]	2017	RGB-D	-	-	-	-	-	-	19.46	19.4	-	-	-	-	-
		Garon [42]	2018	RGB-D	-	-	-	-	-	-	**12.39**	**8.11**	-	-	-	-	-
		DeepIM [119]	2018	RGB	-	-	-	-	-	-	-	-	79.3	91.0	-	-	83.3
		se(3)-TrackNet [43]	2020	RGB-D	-	-	-	-	-	-	-	-	93.0	95.7	92.6	95.5	11.0
		HFF6D [124]	2022	RGB-D	-	-	-	-	-	-	-	-	-	95.6	92.88	95.80	3.3
		FCR-TrackNet [125]	2023	RGB-D	-	-	-	-	-	-	-	-	94.5	96.7	**93.2**	96.0	11.2
		FoundationPose [23]	2024	RGB-D	-	-	-	-	-	-	-	-	**96.0**	**97.9**	93.09	**96.42**	31.25
Filter-based		Wüthrich [138]	2013	Depth	-	-	-	-	-	-	-	-	78.0	90.2	78.3	89.2	77.5
		Choi [59]	2013	RGB-D	-	-	-	-	1.36	2.45	-	-	-	-	-	-	132
		RGF [142]	2016	Depth	-	-	-	-	-	-	-	-	59.2	74.3	30.0	40.0	84.7
Hybrid	LF	Krull [57]	2015	RGB-D	-	-	-	-	0.82	1.38	-	-	-	-	-	-	131
		PoseRBPF [45]	2019	RGB-D	-	-	-	-	-	-	-	-	80.8	93.3	-	-	200
	LO	Tan17 [65]	2017	Depth	-	-	-	-	0.10	0.07	-	-	-	-	-	-	2.2
		DeepAC[144]	2023	RGB	16.67	93.3	94	87.2	-	-	-	-	-	-	-	-	133.7

2) Classical filtering approaches (2013–2016) achieve modest accuracy but suffer from high runtimes (70~130 ms), making them less practical for real-time tracking.
3) For hybrid methods, PoseRBPF [45] shows strong accuracy on YCB-Video (greater than 90%) but with a very high runtime of approximately 200 ms. DeepAC [144] combines learning and optimization, achieving competitive accuracy (around 94% on RBOT) but still requiring heavy computation of about 134 ms.
4) Early Random Forest (RF) methods (2014–2016) had limited performance. Deep learning-based methods dominate from 2017 onward. Deep-learning-based methods have demonstrated strong performance across multiple datasets. FoundationPose [23] achieves state-of-the-art performance across YCB-Video and YCBInEOAT, though runtime is relatively higher (~31 ms). FCR-TrackNet [125] also performs strongly, with over 93% accuracy and moderate runtime (~11 ms).
5) FoundationPose achieves state-of-the-art results on novel objects without relying on object-specific training data. This highlights how the combination of large-scale synthetic training and transformer-based architectures can enable robust and efficient 6D pose estimation across diverse and unseen scenarios.

6. Applications

In this section, we discuss key application domains that build upon advances in instance-level 6D pose tracking, focusing on robotic manipulation, augmented reality (AR)/virtual reality (VR), and human-robot interaction (HRI).

6.1. Robotic manipulation

In robotics, accurate 6D object pose tracking is a core perceptual skill for enabling physical interaction. While single-frame pose estimation provides initial localization, continuous tracking is essential for closed-loop control, allowing robots to adapt in real time to changes in an object's motion or state. Applications range from dynamic grasping to high-precision assembly, both of which critically depend on robust pose tracking.

Burgess et al. [147] apply position-based visual servoing (PBVS) to dynamic grasping, where robots must intercept and grasp objects with unpredictable trajectories. Using an eye-in-hand camera, the object's 6D pose is continuously tracked, and control commands are computed in task space to guide the robot's motion. While effective, traditional PBVS methods often struggle under rapid or erratic motion, motivating more advanced approaches such as [148], which integrates a deep-learning-based pose tracker with Kalman filtering and reinforcement learning to achieve greater robustness in dynamic scenes.

Another application domain is high-precision assembly, where tasks like peg-in-hole or plug insertion demand sub-millimeter accuracy, placing strict requirements on pose tracker accuracy. To mitigate minor perception errors, Wen [149] demonstrates that combining a real-time RGB-D 6D pose tracker with a compliant manipulator enables natural error correction. This approach achieves reliable insertions without relying on costly force sensors or ultra-precise hardware.

6.2. Augmented reality/Virtual reality

AR/VR systems require continuous, real-time, and accurate 6DoF pose tracking of manipulated objects to seamlessly fuse virtual and real elements, creating immersive experiences. A concrete example is AR assembly guidance in intelligent manufacturing, where optical see-through head-mounted displays overlay 3D virtual information on physical parts to provide intuitive visual feedback that enhances efficiency and decision-making. Chen et al. [107] demonstrate this by integrating a 6D pose tracker into a custom AR headset with a camera module and see-through display. The system tracks texture-less industrial parts in real time, estimates their pose relative to the target assembly position, and guides operators by 2D/3D rendering.

Medical AR enables immersive education and training by tracking surgical instruments and anatomical models, allowing students and clinicians to practice complex procedures in a safe, simulated environment with interactive 3D guidance. For example, AR can overlay virtual anatomical structures directly on a physical mannequin or patient, helping trainees visualize hidden organs or blood vessels while rehearsing surgical steps. Consumer-facing AR applications also rely on object tracking, from furniture visualization that anchors 3D models in a home environment to gaming and entertainment, where physical objects are transformed into interactive props. These consumer applications demand trackers that run in real time on mobile devices with limited computational resources, while still ensuring rendering stability and low latency to preserve immersion.

6.3. Human-Robot Interaction

In Human-Robot Interaction (HRI), particularly in collaborative assembly tasks, instance-level 6D pose tracking provides the perceptual backbone that transforms a robot from a passive tool into an intelligent and adaptive partner. Beyond simply localizing objects, the tracker is leveraged to infer the broader assembly state and task context, enabling the robot to reason about human actions and determine how best to assist.

Hadfield et al. [150] demonstrate Object Assembly Guidance in child-robot educational scenarios. A depth sensor was used to track the 6D poses of individual parts as children attempted to build structures. By analyzing the relative poses of the components, the system inferred the current assembly state and enabled a humanoid robot to provide real-time feedback in the form of verbal instructions, pointing gestures, and encouragement, effectively guiding children through the task. In [151], the robot collaborates with a human operator on the same assembly task. Here, 6D poses from the tracker help extract task context by identifying the current step in the assembly sequence, while also verifying action outcomes by confirming whether a part has been correctly placed or manipulated by either the human or the robot. This continuous flow of perceptual information allows the collaborative system to perceive, reason, adapt, and plan within an integrated cognitive architecture.

7. Future directions and challenges

This work has introduced the latest developments in the field of instance-level 6D object pose tracking. To further advance this research area, this section analyzes and introduces several potential future research directions and challenges.

7.1. Enhanced robustness in complex scenes

While existing algorithms achieve strong results in controlled environments, their performance often degrades in complex real-world settings. In AR assembly guidance, for example, the tracked objects are frequently texture-less industrial parts, which are difficult to distinguish from other texture-less assembly parts in the backgrounds. Additional challenges include strong occlusions caused by hands or tools during manipulation and motion blur from rapid movements or camera shake. Similar issues arise in robotic bin-picking, where objects are stacked closely together, leading to severe mutual occlusion and confusion from visually similar distractors. These factors generate noisy visual cues that can mislead both classical and learning-based methods. The degradation of pose tracking performance is evident in the industrial texture-less InOT6D dataset, where most methods perform significantly worse on the more challenging variants, including Unmodeled Occlusion (UOC), Motion Blur (MOT), and Color Ambiguity (COR).

To address these challenges, future research should focus on models capable of integrating complementary cues. Classical methods such as FSCGE [63], which rely on multi-feature fusion to reduce dependence on single cues, demonstrate the promise of this direction. However, how to design optimal fusion strategies remains an open question. Multi-sensor fusion also offers promising opportunities: multi-view camera setups can alleviate occlusion sensitivity, while event cameras can provide blur-free perception for high-speed motion. Dubeau et al. [152] demonstrate how adding event camera data help to reduce tracking error and tracking lost when objects move fast. Beyond visual sensing, incorporating additional modalities shows strong potential. For instance, Li et al. [153] propose V-HOP, which combines visual and haptic input to enhance robustness in challenging occlusion scenarios, underscoring the advantages of multimodal perception for reliable instance-level 6DoF tracking.

7.2. Handling diverse material properties

Most existing datasets and methods focus on opaque, diffusely reflecting objects. However, in industrial automation, transparent, reflective, and metallic objects are common, for example in metal sheet processing or glassware production. Metallic surfaces are difficult to handle due to strong specular reflections, while transparent objects present an even greater challenge: instead of revealing their own texture, RGB observations often capture only a refracted view of the background. In such cases, a traditional textured CAD model is insufficient. Emerging techniques such as neural radiance fields (NeRF) and Gaussian splatting offer promising solutions by learning continuous 3D representations of objects along with their light-field properties directly from images. These approaches enable more realistic rendering and improve robustness when dealing with untextured surfaces and specular highlights. One example is BundleSDF [154] , which simultaneously performs 6DoF tracking and neural 3D reconstruction of unknown objects from monocular RGB-D video, highlighting the potential of neural implicit representations for handling challenging materials.

7.3. Exploring more temporal information

Existing deep-learning-based methods typically rely on only two frames, the current frame and the immediate previous frame, to estimate object poses. This limited temporal scope fails to utilize valuable information from earlier frames, which can be critical in challenging scenarios such as heavy occlusion or cluttered backgrounds. Leveraging a longer temporal context holds great potential for enhancing pose tracking performance under such conditions. One promising direction is to integrate deep neural networks within a filtering framework, enabling the model to exploit temporal consistency and smooth pose predictions over time. Another promising approach is to design a memory-augmented network that encodes and stores features from multiple past frames in a learnable memory bank. This allows the model to attend to and retrieve relevant historical cues when predicting the current pose, effectively combining current observations with rich temporal context. For example, Tan et al. [155] propose a framework that employs transformers for joint spatial-temporal reasoning across video sequences. Their approach captures both spatial relationships within each frame and temporal dependencies across consecutive frames, enabling the aggregation of information over time and yielding more accurate and robust pose predictions.

7.4. Pose uncertainty estimation

Many existing methods do not explicitly incorporate pose uncertainty into their algorithm design. For evaluation on datasets, most approaches initialize tracking only at the first frame, without considering tracking reliability over time. Some methods apply simple heuristics, such as declaring tracking failure when the pose change between consecutive frames exceeds a predefined threshold. However, such heuristics are often unreliable and can lead to false alarms or missed failures. In real-world applications that demand high reliability, the ability to accurately detect tracking failures and trigger timely re-initialization is essential. For example, in human-robot collaborative assembly, objects may frequently leave the camera's field of view due to human occlusion or unpredictable behavior. A tracker that can detect such failures and re-localize the object once it reappears enhances the robot's understanding of human actions, enabling it to provide more effective assistance.

FoundationPose [23] introduced a pose selection module that predicts scores for multiple pose hypotheses, but this scoring is designed for relative ranking rather than explicit failure detection. Its effectiveness for tracking failure detection remains unclear. Guo et al. proposed RGBTrack [156], which combines a state machine with a 2D object tracker (e.g., XMem [157] or SAM2 [158]) and a Kalman filter to monitor tracking status. While these efforts show promise, they remain preliminary. Developing reliable mechanisms for detecting tracking failures and enabling fast recovery remains an important open problem for future research.

7.5. Domain gap reduction and generalization

Most deep learning-based approaches rely heavily on synthetic datasets for training, making the reduction of the synthetic-to-real gap a key research challenge. Traditional strategies such as Domain Randomization and Physically Based Rendering (PBR) have shown strong potential in addressing this issue. The latest trend is to leverage advances in diffusion models and generative AI to synthesize highly realistic and diverse training data. Unlike conventional rendering pipelines, generative methods can reproduce complex visual effects such as transparency, specular reflections, and subtle lighting variations that are difficult to simulate with graphics-based approaches. Tools like ControlNet [159] further enhance this process by enabling conditional generation based on structural inputs (e.g., edges or depth maps). A promising direction is to use PBR-generated data as guidance for generative rendering, combining the structure realism of physics-based simulation with the texture-realism and diversity of diffusion-based generation.

Foundation models such as FoundationPose [23], trained on massive and diverse datasets, have demonstrated remarkable generalization to unseen objects. However, for tasks requiring maximum accuracy, such as precise robotic assembly, fine-tuning with instance-specific data may provide an additional performance boost. Such data can capture subtle geometric and material properties of specific objects that a general model might overlook. How to efficiently leverage large foundation models while fine-tuning them for instance-specific tracking remains an important open question for future research.

7.6. Computational efficiency and real-time performance

Many applications, such as AR on mobile devices, require real-time processing under limited computational resources. While deep learning-based methods achieve high accuracy, they typically rely on powerful GPUs for inference, making deployment on resource-constrained platforms challenging. To achieve real-time performance, it is essential to adopt lightweight network architectures and apply post-training optimization techniques, such as model compression, pruning, quantization, and knowledge distillation, to lower computational cost and memory usage. Beyond algorithmic improvements, efficiency is also constrained by hardware. Edge computing offers a promising solution by offloading heavy computation to nearby servers, enabling real-time performance even on devices with limited capacity.

7.7. Unified benchmarking framework

Although several datasets exist for instance-level 6D object pose tracking, they are fragmented, with each one addressing only a subset of real-world challenges. This leads to new methods being evaluated on different datasets, making fair and direct performance comparisons difficult. To ensure consistent and reproducible evaluation, the field requires a unified benchmarking framework for 6D pose tracking, analogous to the successful BOP benchmark [29] for 6D pose estimation. While BOP is a cornerstone for estimation, many of its datasets cannot be directly used for tracking evaluation due to their lack of consecutive video frames.

A future tracking benchmark should include a diverse collection of video sequences that capture a broad range of real-world conditions. In addition to widely used datasets, it should also integrate datasets that target specific challenges. For example, Fast-YCB [47] focuses on fast object motions, HO-3D [160] provides complex hand-object interaction sequences with challenging rotational motions, and ClearPose [161] addresses difficult materials by offering large-scale data for transparent objects. Alongside diverse datasets, a standardized evaluation protocol is essential for meaningful comparisons. This protocol should extend beyond common accuracy metrics to address aspects unique to tracking. A critical element is the handling of initialization and re-initialization. Key criteria to specify and report include: whether re-initialization is permitted, the conditions that trigger a reset (e.g., failure thresholds), and the number of resets performed by each method.

8. Conclusion

This paper presents a comprehensive review of instance-level 6D object pose tracking methods. These methods are categorized based on their problem modeling approaches, providing a clear and discriminative classification. Evaluation metrics and common datasets are introduced in detail. Additionally, a quantitative performance comparison of these methods is provided. Key application domains are then discussed, and finally, future research directions and open challenges are analyzed.

References

[1] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, J. Xiao, Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge, in: 2017 IEEE international conference on robotics and automation (ICRA), IEEE, 2017, pp. 1386–1383.

[2] A. Collet, M. Martinez, S. S. Srinivasa, The moped framework: Object recognition and pose estimation for manipulation, The international journal of robotics research 30 (10) (2011) 1284–1306.

[3] C. Choi, H. I. Christensen, Real-time 3d model-based tracking using edge and keypoint features for robotic manipulation, in: 2010 IEEE international conference on robotics and automation, IEEE, 2010, pp. 4048–4055.

[4] H. Wuest, D. Stricker, Tracking of industrial objects by using cad models, JVRB-Journal of Virtual Reality and Broadcasting 4 (1) (2007).

[5] G. Reitmayr, T. W. Drummond, Going out: robust model-based tracking for outdoor augmented reality, in: 2006 IEEE/ACM international symposium on mixed and augmented reality, IEEE, 2006, pp. 109–118.

[6] E. Marchand, H. Uchiyama, F. Spindler, Pose estimation for augmented reality: a hands-on survey, IEEE transactions on visualization and computer graphics 22 (12) (2015) 2633–2651.

[7] R. Held, A. Gupta, B. Curless, M. Agrawala, 3d puppetry: a kinect-based interface for 3d animation., in: UIST, Vol. 12, 2012, pp. 423–434.

[8] P.-C. Wu, R. Wang, K. Kin, C. Twigg, S. Han, M.-H. Yang, S.-Y. Chien, Dodecapen: Accurate 6dof tracking of a passive stylus, in: Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, 2017, pp. 365–374.

[9] B. Drost, M. Ulrich, N. Navab, S. Ilic, Model globally, match locally: Efficient and robust 3d object recognition, in: 2010 IEEE computer society conference on computer vision and pattern recognition, Ieee, 2010, pp. 998–1005.

[10] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, N. Navab, Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes, in: Asian conference on computer vision, Springer, 2012, pp. 548–562.

[11] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, N. Navab, Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 1521–1529.

[12] Y. Xiang, T. Schmidt, V. Narayanan, D. Fox, Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes, arXiv preprint arXiv:1711.00199 (2017).

[13] K. Park, T. Patten, M. Vincze, Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 7668–7677.

[14] Z. Li, G. Wang, X. Ji, Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 7678–7687.

[15] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, J. Sun, Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 11632–11641.

[16] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, R. Triebel, Implicit 3d orientation learning for 6d object detection from rgb images, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 699–715.

[17] C. Wang, R. Martín-Martín, D. Xu, J. Lv, C. Lu, L. Fei-Fei, S. Savarese, Y. Zhu, 6-pack: Category-level 6d pose tracker with anchor-based keypoints, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2020, pp. 10059–10066.

[18] Y. Weng, H. Wang, Q. Zhou, Y. Qin, Y. Duan, Q. Fan, B. Chen, H. Su, L. J. Guibas, Captra: Category-level pose tracking for rigid and articulated objects from point clouds, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 13209–13218.

[19] S. Yu, D.-H. Zhai, Y. Xia, D. Li, S. Zhao, Cattrack: Single-stage category-level 6d object pose tracking via convolution and vision transformer, IEEE Transactions on Multimedia 26 (2023) 1665–1680.

[20] X. Cao, J. Li, P. Zhao, J. Li, X. Qin, Corr-track: Category-level 6d pose tracking with soft-correspondence matrix estimation, IEEE Transactions on Visualization and Computer Graphics (2024).

[21] P. Ausserlechner, D. Haberger, S. Thalhammer, J.-B. Weibel, M. Vincze, Zs6d: Zero-shot 6d object pose estimation using vision transformers, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024, pp. 463–469.

[22] Y. Labbé, L. Manuelli, A. Mousavian, S. Tyree, S. Birchfield, J. Tremblay, J. Carpentier, M. Aubry, D. Fox, J. Sivic, Megapose: 6d pose estimation of novel objects via render & compare, arXiv preprint arXiv:2212.06870 (2022).

[23] B. Wen, W. Yang, J. Kautz, S. Birchfield, Foundationpose: Unified 6d pose estimation and tracking of novel objects, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 17868–17879.

[24] C. Sahin, G. Garcia-Hernando, J. Sock, T.-K. Kim, A review on object pose recovery: from 3d bounding box detectors to full 6d pose estimators, Image and Vision Computing (2020) 103898.

[25] J. Guan, Y. Hao, Q. Wu, S. Li, Y. Fang, A survey of 6dof object pose estimation methods for different application scenarios, Sensors 24 (4) (2024) 1076.

[26] J. Liu, W. Sun, H. Yang, Z. Zeng, C. Liu, J. Zheng, X. Liu, H. Rahmani, N. Sebe, A. Mian, Deep learning-based object pose estimation: A comprehensive survey, arXiv preprint arXiv:2405.07801 (2024).

[27] J. Wang, G. Liu, W. Ding, Y. Li, W. Song, From visual understanding to 6d pose reconstruction: A cutting-edge review of deep learning-based object pose estimation, Displays (2025) 103069.

[28] A. V. Patil, P. Rabha, A survey on joint object detection and pose estimation using monocular vision, arXiv preprint arXiv:1811.10216 (2018).

[29] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, J. Matas, BOP challenge 2020 on 6D object localization, European Conference on Computer Vision Workshops (ECCVW) (2020).

[30] L. Vacchetti, V. Lepetit, P. Fua, Stable real-time 3d tracking using online and offline information, IEEE transactions on pattern analysis and machine intelligence 26 (10) (2004) 1385–1391.

[31] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, D. Schmalstieg, Real-time detection and tracking for augmented reality on mobile phones, IEEE transactions on visualization and computer graphics 16 (3) (2009) 355–368.

[32] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International journal of computer vision 60 (2) (2004) 91–110.

[33] C. Harris, C. Stennett, Rapid-a video rate object tracker., in: BMVC, 1990, pp. 1–6.

[34] T. Drummond, R. Cipolla, Real-time visual tracking of complex structures, IEEE Transactions on pattern analysis and machine intelligence 24 (7) (2002) 932–946.

[35] A. I. Comport, E. Marchand, M. Pressigout, F. Chaumette, Real-time markerless tracking for augmented reality: the virtual visual servoing framework, IEEE Transactions on visualization and computer graphics 12 (4) (2006) 615–628.

[36] V. A. Prisacariu, I. D. Reid, Pwp3d: Real-time segmentation and tracking of 3d objects, International journal of computer vision 98 (3) (2012) 335–354.

[37] H. Tjaden, U. Schwanecke, E. Schömer, D. Cremers, A region-based gauss-newton approach to real-time monocular multiple object tracking, IEEE transactions on pattern analysis and machine intelligence 41 (8) (2018) 1797–1812.

[38] M. Stoiber, M. Pfanne, K. H. Strobl, R. Triebel, A. Albu-Schäffer, Srt3d: A sparse region-based 3d object tracking approach for the real world, International Journal of Computer Vision 130 (4) (2022) 1008–1030.

[39] A. Crivellaro, V. Lepetit, Robust 3d tracking with descriptor fields, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3414–3421.

[40] B.-K. Seo, H. Wuest, A direct method for robust model-based 3d object tracking from a monocular rgb image, in: European conference on computer vision, Springer, 2016, pp. 551–562.

[41] M. Garon, J.-F. Lalonde, Deep 6-dof tracking, IEEE transactions on visualization and computer graphics 23 (11) (2017) 2410–2418.

[42] M. Garon, D. Laurendeau, J.-F. Lalonde, A framework for evaluating 6-dof object trackers, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 582–597.

[43] B. Wen, C. Mitash, B. Ren, K. E. Bekris, se (3)-tracknet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains, arXiv preprint arXiv:2007.13866 (2020).

[44] J. M. Wong, V. Kee, T. Le, S. Wagner, G.-L. Mariottini, A. Schneider, L. Hamilton, R. Chipalkatty, M. Hebert, D. M. Johnson, et al., Segicp: Integrated deep semantic segmentation and pose estimation, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2017, pp. 5784–5789.

[45] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, D. Fox, Poserbpf: A rao-blackwellized particle filter for 6d object pose tracking, arXiv preprint arXiv:1905.09304 (2019).

[46] N. A. Piga, F. Bottarel, C. Fantacci, G. Vezzani, U. Pattacini, L. Natale, Maskukf: An instance segmentation aided unscented kalman filter for 6d object pose and velocity tracking, Frontiers in Robotics and AI 8 (2021) 38.

[47] N. A. Piga, Y. Onyshchuk, G. Pasquale, U. Pattacini, L. Natale, Roft: Real-time optical flow-aided 6d object pose and velocity tracking, IEEE Robotics and Automation Letters 7 (1) (2021) 159–166.

[48] V. Lepetit, P. Fua, Monocular model-based 3D tracking of rigid objects, Now Publishers Inc, 2005.

[49] P. Han, G. Zhao, A review of edge-based 3d tracking of rigid objects, Virtual Reality & Intelligent Hardware 1 (6) (2019) 580–596.

[50] Z. Fan, Y. Zhu, Y. He, Q. Sun, H. Liu, J. He, Deep learning on monocular object pose detection and tracking: A comprehensive overview, ACM Computing Surveys 55 (4) (2022) 1–40.

[51] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, C. Rother, Learning 6d object pose estimation using 3d object coordinates, in: European conference on computer vision, Springer, 2014, pp. 536–551.

[52] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, T.-K. Kim, Recovering 6d object pose and predicting next-best-view in the crowd, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 3583–3592.

[53] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, X. Zabulis, T-less: An rgb-d dataset for 6d pose estimation of texture-less objects, in: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2017, pp. 880–888.

[54] C. Rennie, R. Shome, K. E. Bekris, A. F. De Souza, A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place, IEEE Robotics and Automation Letters 1 (2) (2016) 1179–1185.

[55] A. Tejani, D. Tang, R. Kouskouridas, T.-K. Kim, Latent-class hough forests for 3d object detection and pose estimation, in: European Conference on Computer Vision, Springer, 2014, pp. 462–477.

[56] T. Hodaň, F. Michel, E. Brachmann, W. Kehl, A. Glent Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas, C. Rother, BOP: Benchmark for 6D object pose estimation, European Conference on Computer Vision (ECCV) (2018).

[57] A. Krull, F. Michel, E. Brachmann, S. Gumhold, S. Ihrke, C. Rother, 6-dof model based tracking via object coordinate regression, in: Asian Conference on Computer Vision, Springer, 2014, pp. 384–399.

[58] S. Akkaladevi, M. Ankerl, C. Heindl, A. Pichler, Tracking multiple rigid symmetric and non-symmetric objects in real-time using depth data, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2016, pp. 5644–5649.

[59] C. Choi, H. I. Christensen, Rgb-d object tracking: A particle filter approach on gpu, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 1084–1091.

[60] G. Wang, B. Wang, F. Zhong, X. Qin, B. Chen, Global optimal searching for textureless 3d object tracking, The Visual Computer 31 (2015) 979–988.

[61] P.-C. Wu, Y.-Y. Lee, H.-Y. Tseng, H.-I. Ho, M.-H. Yang, S.-Y. Chien, [poster] a benchmark dataset for 6dof object pose tracking, in: 2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct), IEEE, 2017, pp. 186–191.

[62] J. Li, B. Wang, S. Zhu, X. Cao, F. Zhong, W. Chen, T. Li, J. Gu, X. Qin, Bcot: A markerless high-precision 3d object tracking benchmark, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 6697–6706.

[63] N. Lv, D. Zhao, F. Kong, Z. Xu, F. Du, A multi-feature fusion-based pose tracking method for industrial object with visual ambiguities, Advanced Engineering Informatics 62 (2024) 102788.

[64] D. J. Tan, S. Ilic, Multi-forest tracker: A chameleon in tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1202–1209.

[65] D. J. Tan, N. Navab, F. Tombari, Looking beyond the simple scenarios: Combining learners and optimizers in 3d temporal tracking, IEEE transactions on visualization and computer graphics 23 (11) (2017) 2399–2409.

[66] W. Kehl, F. Tombari, S. Ilic, N. Navab, Real-time 3d model tracking in color and depth on a single cpu core, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 745–753.

[67] M. Stoiber, M. Sundermeyer, R. Triebel, Iterative corresponding geometry: Fusing region and depth for highly efficient 3d tracking of textureless objects, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 6855–6865.

[68] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, A. M. Dollar, The ycb object and model set: Towards common benchmarks for manipulation research, in: 2015 international conference on advanced robotics (ICAR), IEEE, 2015, pp. 510–517.

[69] B. Drost, M. Ulrich, P. Bergmann, P. Härtinger, C. Steger, Introducing mvtec itodd — a dataset for 3d object recognition in industry, in: 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), 2017, pp. 2200–2208. doi:10.1109/ICCVW.2017.257.

[70] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, Orb: An efficient alternative to sift or surf, in: 2011 International conference on computer vision, Ieee, 2011, pp. 2564–2571.

[71] S. Leutenegger, M. Chli, R. Y. Siegwart, Brisk: Binary robust invariant scalable keypoints, in: 2011 International conference on computer vision, Ieee, 2011, pp. 2548–2555.

[72] I. Skrypnyk, D. G. Lowe, Scene modelling, recognition and tracking with invariant image features, in: Third IEEE and ACM international symposium on mixed and augmented reality, IEEE, 2004, pp. 110–119.

[73] A. Ladikos, S. Benhimane, N. Navab, A real-time tracking system combining template-based and feature-based approaches., in: VISAPP (2), 2007, pp. 325–332.

[74] K. Kim, V. Lepetit, W. Woo, Keyframe-based modeling and tracking of multiple 3d objects, in: 2010 IEEE International Symposium on Mixed and Augmented Reality, IEEE, 2010, pp. 193–198.

[75] B. Lukas, An iterative image registration technique with an application to stereo vision, in: Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI), 1981, 1981, pp. 674–679.

[76] C. Tomasi, T. Kanade, Detection and tracking of point, Int J Comput Vis 9 (137-154) (1991) 2.

[77] J. Shi, et al., Good features to track, in: 1994 Proceedings of IEEE conference on computer vision and pattern recognition, IEEE, 1994, pp. 593–600.

[78] B. K. Horn, B. G. Schunck, Determining optical flow, Artificial intelligence 17 (1-3) (1981) 185–203.

[79] Y. Park, V. Lepetit, W. Woo, Extended keyframe detection with stable tracking for multiple 3d object tracking, IEEE Transactions on Visualization and Computer Graphics 17 (11) (2010) 1728–1735.

[80] I. E. Sutherland, Three-dimensional data input by tablet, Proceedings of the IEEE 62 (4) (1974) 453–461.

[81] V. Lepetit, F. Moreno-Noguer, P. Fua, Epnp: Efficient perspective-n-point camera pose estimation, Int. J. Comput. Vis 81 (2) (2009) 155–166.

[82] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge university press, 2003.

[83] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (6) (1981) 381–395.

[84] K. W. Chia, A. D. Cheok, S. J. Prince, Online 6 dof augmented reality registration from natural features, in: Proceedings. International Symposium on Mixed and Augmented Reality, IEEE, 2002, pp. 305–313.

[85] Y. Park, V. Lepetit, W. Woo, Multiple 3d object tracking for augmented reality, in: 2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, IEEE, 2008, pp. 117–120.

[86] E. Marchand, P. Bouthemy, F. Chaumette, A 2d–3d model-based approach to real-time visual tracking, Image and Vision Computing 19 (13) (2001) 941–955.

[87] L. Vacchetti, V. Lepetit, P. Fua, Combining edge and texture information for real-time accurate 3d camera tracking, in: Third IEEE and ACM international symposium on mixed and augmented reality, IEEE, 2004, pp. 48–56.

[88] H. Wuest, F. Vial, D. Stricker, Adaptive line tracking with multiple hypotheses for augmented reality, in: Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'05), IEEE, 2005, pp. 62–69.

[89] B.-K. Seo, H. Park, J.-I. Park, S. Hinterstoisser, S. Ilic, Optimal local searching for fast and robust textureless 3d object tracking in highly cluttered backgrounds, IEEE transactions on visualization and computer graphics 20 (1) (2013) 99–110.

[90] M. Imperoli, A. Pretto, D co: Fast and robust registration of 3d textureless objects using the directional chamfer distance, in: International conference on computer vision systems, Springer, 2015, pp. 316–328.

[91] B. Bugaev, A. Kryshchenko, R. Belov, Combining 3d model contour energy and keypoints for object tracking, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 53–69.

[92] B. Wang, F. Zhong, X. Qin, Robust edge-based 3d object tracking with direction-based pose validation, Multimedia Tools and Applications 78 (2019) 12307–12331.

[93] H. Huang, F. Zhong, Y. Sun, X. Qin, An occlusion-aware edge-based method for monocular 3d object tracking using edge confidence, in: Computer Graphics Forum, Vol. 39, Wiley Online Library, 2020, pp. 399–409.

[94] X. Sun, J. Zhou, W. Zhang, Z. Wang, Q. Yu, Robust monocular pose tracking of less-distinct objects based on contour-part model, IEEE Transactions on Circuits and Systems for Video Technology 31 (11) (2021) 4409–4421.

[95] J.-C. Li, F. Zhong, S.-H. Xu, X.-Y. Qin, 3d object tracking with adaptively weighted local bundles, Journal of Computer Science and Technology 36 (3) (2021) 555–571.

[96] J. Li, X. Song, F. Zhong, X. Qin, Fast 3d texture-less object tracking with geometric contour and local region, Computers & Graphics 97 (2021) 225–235.

[97] X. Tian, X. Lin, F. Zhong, X. Qin, Large-displacement 3d object tracking with hybrid non-local optimization, in: European Conference on Computer Vision, Springer, 2022, pp. 627–643.

[98] H. Tjaden, U. Schwanecke, E. Schomer, Real-time monocular pose estimation of 3d objects using temporally consistent local color histograms, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 124–132.

[99] D. Kraft, A software package for sequential quadratic programming, Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt (1988).

[100] V. A. Prisacariu, O. Kähler, D. W. Murray, I. D. Reid, Real-time 3d tracking and reconstruction on mobile phones, IEEE transactions on visualization and computer graphics 21 (5) (2014) 557–570.

[101] S. Zhao, L. Wang, W. Sui, H.-y. Wu, C. Pan, 3d object tracking via boundary constrained region-based model, in: 2014 IEEE International Conference on Image Processing (ICIP), IEEE, 2014, pp. 486–490.

[102] H. Tjaden, U. Schwanecke, E. Schömer, Real-time monocular segmentation and pose tracking of multiple objects, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14, Springer, 2016, pp. 423–438.

[103] J. Hexner, R. R. Hagege, 2d-3d pose estimation of heterogeneous objects using a region based approach, International Journal of Computer Vision 118 (2016) 95–112.

[104] L. Zhong, X. Zhao, Y. Zhang, S. Zhang, L. Zhang, Occlusion-aware region-based 3d pose tracking of objects with temporally consistent polar-based local partitioning, IEEE Transactions on Image Processing 29 (2020) 5065–5078.

[105] H. Huang, F. Zhong, X. Qin, Pixel-wise weighted region-based 3d object tracking using contour constraints, IEEE Transactions on Visualization and Computer Graphics 28 (12) (2021) 4319–4331.

[106] M. Stoiber, M. Elsayed, A. E. Reichert, F. Steidle, D. Lee, R. Triebel, Fusing visual appearance and geometry for multi-modality 6dof object tracking, arXiv preprint arXiv:2302.11458 (2023).

[107] J. Chen, J. Chen, K. Liu, H. Chang, S. Fu, J. Yang, Robust 6dof pose tracking considering contour and interior correspondence uncertainty for ar assembly guidance, IEEE Transactions on Instrumentation and Measurement (2025).

[108] H. Alismail, B. Browning, S. Lucey, Robust tracking in low light and sudden illumination changes, in: 2016 Fourth International Conference on 3D Vision (3DV), IEEE, 2016, pp. 389–398.

[109] L. Chen, F. Zhou, Y. Shen, X. Tian, H. Ling, Y. Chen, Illumination insensitive efficient second-order minimization for planar object tracking, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 4429–4436.

[110] J. Engel, V. Koltun, D. Cremers, Direct sparse odometry, IEEE transactions on pattern analysis and machine intelligence 40 (3) (2017) 611–625.

[111] S. Baker, I. Matthews, Lucas-kanade 20 years on: A unifying framework, International journal of computer vision 56 (2004) 221–255.

[112] E. Malis, Improving vision-based control using efficient second-order minimization techniques, in: IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004, Vol. 2, IEEE, 2004, pp. 1843–1848.

[113] L. Zhong, M. Lu, L. Zhang, A direct 3d object tracking method based on dynamic textured model rendering and extended dense feature fields, IEEE Transactions on Circuits and Systems for Video Technology 28 (9) (2017) 2302–2315.

[114] L. Sevilla-Lara, E. Learned-Miller, Distribution fields for tracking, in: 2012 IEEE Conference on computer vision and pattern recognition, IEEE, 2012, pp. 1910–1917.

[115] R. Zabih, J. Woodfill, Non-parametric local transforms for computing visual correspondence, in: Computer Vision—ECCV'94: Third European Conference on Computer Vision Stockholm, Sweden, May 2–6 1994 Proceedings, Volume II 3, Springer, 1994, pp. 151–158.

[116] G. Caron, A. Dame, E. Marchand, Direct model based visual tracking and pose estimation using mutual information, Image and Vision Computing 32 (1) (2014) 54–63.

[117] G. G. Scandaroli, M. Meilland, R. Richa, Improving ncc-based direct visual tracking, in: Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12, Springer, 2012, pp. 442–455.

[118] D. Joseph Tan, F. Tombari, S. Ilic, N. Navab, A versatile learning-based 3d temporal tracker: Scalable, robust, online, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 693–701.

[119] Y. Li, G. Wang, X. Ji, Y. Xiang, D. Fox, Deepim: Deep iterative matching for 6d pose estimation, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 683–698.

[120] F. Manhardt, W. Kehl, N. Navab, F. Tombari, Deep model-based 6d pose refinement in rgb, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 800–815.

[121] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, S. Savarese, Densefusion: 6d object pose estimation by iterative dense fusion, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 3343–3352.

[122] S. Zakharov, I. Shugurov, S. Ilic, Dpod: 6d pose object detector and refiner, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 1941–1950.

[123] A. Trabelsi, M. Chaabane, N. Blanchard, R. Beveridge, A pose proposal and refinement network for better 6d object pose estimation, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 2382–2391.

[124] J. Liu, W. Sun, C. Liu, X. Zhang, S. Fan, W. Wu, Hff6d: Hierarchical feature fusion network for robust 6d object pose tracking, IEEE Transactions on Circuits and Systems for Video Technology 32 (11) (2022) 7719–7731.

[125] W. Zhu, H. Feng, Y. Yi, M. Zhang, Fcr-tracknet: Towards high-performance 6d pose tracking with multi-level features fusion and joint classification-regression, Image and Vision Computing 135 (2023) 104698.

[126] J. Wang, G. Duan, Y. Wang, G. Yi, L. Dong, Z. Wang, X. Zhang, S. Zhang, Cross-domain fusion and embedded refinement-based 6d object pose tracking on textureless objects, Journal of Intelligent Manufacturing (2024) 1–15.

[127] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer, Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size, arXiv preprint arXiv:1602.07360 (2016).

[128] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, T. Brox, Flownet: Learning optical flow with convolutional networks, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 2758–2766.

[129] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 31, 2017.

[130] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[131] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).

[132] M. Pupilli, A. Calway, Real-time camera tracking using known 3d models and a particle filter, in: 18th International Conference on Pattern Recognition (ICPR'06), Vol. 1, IEEE, 2006, pp. 199–203.

[133] G. Klein, D. W. Murray, Full-3d edge tracking with a particle filter., in: BMVC, 2006, pp. 1119–1128.

[134] Y. Yoon, A. Kosaka, A. C. Kak, A new kalman-filter-based framework for fast and accurate visual tracking of rigid objects, IEEE Transactions on Robotics 24 (5) (2008) 1238–1251.

[135] C. Teulière, E. Marchand, L. Eck, Using multiple hypothesis in model-based tracking, in: 2010 IEEE International Conference on Robotics and Automation, IEEE, 2010, pp. 4559–4565.

[136] R. B. Rusu, S. Cousins, 3d is here: Point cloud library (pcl), in: 2011 IEEE international conference on robotics and automation, IEEE, 2011, pp. 1–4.

[137] C. Choi, H. I. Christensen, Robust 3d visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features, The International Journal of Robotics Research 31 (4) (2012) 498–519.

[138] M. Wüthrich, P. Pastor, M. Kalakrishnan, J. Bohg, S. Schaal, Probabilistic object tracking using a range camera, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 3195–3202.

[139] J. Issac, M. Wüthrich, C. G. Cifuentes, J. Bohg, S. Trimpe, S. Schaal, Depth-based object tracking using a robust gaussian filter, in: 2016 IEEE international conference on robotics and automation (ICRA), IEEE, 2016, pp. 608–615.

[140] J.-S. Gutmann, D. Fox, An experimental comparison of localization methods continued, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 1, IEEE, 2002, pp. 454–459.

[141] K. Murphy, S. Russell, Rao-blackwellised particle filtering for dynamic bayesian networks, in: Sequential Monte Carlo methods in practice, Springer, 2001, pp. 499–515.

[142] M. Wüthrich, C. G. Cifuentes, S. Trimpe, F. Meier, J. Bohg, J. Issac, S. Schaal, Robust gaussian filtering using a pseudo measurement, in: 2016 American Control Conference (ACC), IEEE, 2016, pp. 3606–3613.

[143] L. Zhong, Y. Zhang, H. Zhao, A. Chang, W. Xiang, S. Zhang, L. Zhang, Seeing through the occluders: Robust monocular 6-dof object pose tracking via model-guided video object segmentation, IEEE Robotics and Automation Letters 5 (4) (2020) 5159–5166.

[144] L. Wang, S. Yan, J. Zhen, Y. Liu, M. Zhang, G. Zhang, X. Zhou, Deep active contours for real-time 6-dof object tracking, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 14034–14044.

[145] Y. Wu, P. Vandewalle, P. Slaets, E. Demeester, An improved approach to 6d object pose tracking in fast motion scenarios, in: 2022 Sixth IEEE International Conference on Robotic Computing (IRC), IEEE, 2022, pp. 229–237.

[146] M. Stoiber, M. Pfanne, K. H. Strobl, R. Triebel, A. Albu-Schäffer, A sparse gaussian approach to region-based 6dof object tracking, in: Proceedings of the Asian Conference on Computer Vision, 2020.

[147] B. Burgess-Limerick, C. Lehnert, J. Leitner, P. Corke, An architecture for reactive mobile manipulation on-the-move, arXiv preprint arXiv:2212.06991 (2022).

[148] K. Boyalakuntla, A. Boularias, J. Yu, Karl: Kalman-filter assisted reinforcement learner for dynamic object tracking and grasping, arXiv preprint arXiv:2506.15945 (2025).

[149] B. Wen, Beyond instance-level reasoning in object pose estimation and tracking for robotic manipulation, Ph.D. thesis, Rutgers The State University of New Jersey, School of Graduate Studies (2022).

[150] J. Hadfield, P. Koutras, N. Efthymiou, G. Potamianos, C. S. Tzafestas, P. Maragos, Object assembly guidance in child-robot interaction using rgb-d based 3d tracking, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 347–354.

[151] S. C. Akkaladevi, M. Plasch, C. Eitzinger, A. Pichler, B. Rinner, Towards a context enhanced framework for multi object tracking in human robot collaboration, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 168–173.

[152] E. Dubeau, M. Garon, B. Debaque, R. de Charette, J.-F. Lalonde, Rgb-de: Event camera calibration for fast 6-dof object tracking, in: 2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), IEEE, 2020, pp. 127–135.

[153] H. Li, M. Jia, T. Akbulut, Y. Xiang, G. Konidaris, S. Sridhar, V-hop: Visuo-haptic 6d object pose tracking, arXiv preprint arXiv:2502.17434 (2025).

[154] B. Wen, J. Tremblay, V. Blukis, S. Tyree, T. Müller, A. Evans, D. Fox, J. Kautz, S. Birchfield, Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 606–617.

[155] P. X. Tan, D.-C. Hoang, E. Kamioka, A.-N. Nguyen, D.-T. Tran, V.-H. Duong, A.-T. Mai, D.-L. Pham, K.-T. Phan, X.-T. Dinh, et al., Visibility aware in-hand object pose tracking in videos with transformers, IEEE Access (2025).

[156] T. Guo, J. Yu, Rgbtrack: Fast, robust depth-free 6d pose estimation and tracking, arXiv preprint arXiv:2506.17119 (2025).

[157] H. K. Cheng, A. G. Schwing, Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model, in: European conference on computer vision, Springer, 2022, pp. 640–658.

[158] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, C. Feichtenhofer, Sam 2: Segment anything in images and videos, arXiv preprint arXiv:2408.00714 (2024).
URL `https://arxiv.org/abs/2408.00714`

[159] L. Zhang, A. Rao, M. Agrawala, Adding conditional control to text-to-image diffusion models.

[160] S. Hampali, M. Rad, M. Oberweger, V. Lepetit, Honnotate: A method for 3d annotation of hand and object poses, in: CVPR, 2020.

[161] X. Chen, H. Zhang, Z. Yu, A. Opipari, O. Chadwicke Jenkins, Clearpose: Large-scale transparent object dataset and benchmark, in: European conference on computer vision, Springer, 2022, pp. 381–396.