Deep Active Contours for Real-time 6-DoF Object Tracking

Long Wang[1*] Shen Yan[2*] Jianan Zhen[1] Yu Liu[2]
Maojun Zhang[2] Guofeng Zhang[3] Xiaowei Zhou[3†]
[1]SenseTime Research [2]National University of Defense Technology [3]Zhejiang University

Abstract

This paper solves the problem of real-time 6-DoF object tracking from an RGB video. Prior optimization-based methods optimize the object pose by aligning the projected model to the image based on handcrafted features, which are prone to suboptimal solutions. Recent learning-based methods use neural networks to predict the pose, which suffer from limited generalizability or computational efficiency. We propose a learning-based active contour model to make the best use of both worlds. Specifically, given an initial pose, we project the object model to the image plane to obtain the initial contour and use a lightweight network to predict how the contour should move to match the true object boundary, which provides the gradients to optimize the object pose. We also devise an efficient optimization algorithm to train our model end-to-end with pose supervision. Experimental results on semi-synthetic and real-world 6-DoF object tracking datasets demonstrate that our model outperforms state-of-the-art methods by a substantial margin in pose accuracy, while achieving real-time performance on mobile devices. Code is available on our project page: `https://zju3dv.github.io/deep_ac/`.

1. Introduction

Video-based 6-DoF object tracking is the task of tracking the pose of a rigid object from an RGB image sequence, given a predefined object CAD model and an initial pose in the first frame, which has a broad range of applications from augmented reality and robotic manipulation to human-computer interaction. These applications require the tracking algorithms to be real-time and avoid the need for object-specific training.

The predominant methods for 6-DoF object tracking optimize the object pose based on keypoint, edge, or region-based features. Keypoint-based methods [34, 42, 18, 17, 43, 27] involve matching keypoints between a 2D image and a 3D real model. Keypoint features such as SIFT [25], ORB [30], or BRISK [19] have been widely employed in such tasks. Nevertheless, the reliance on rich texture narrows down the applicability of these methods. Instead, edge-based tracking methods [6, 33, 53, 32] rely on edges (explicit or implicit) to calculate the relative pose between two consecutive images. Unfortunately, these approaches face challenges when dealing with background clutter and motion blur, thus limiting their effectiveness. To solve this issue, more recent edge-based methods [46, 44, 14, 13, 39] further incorporate local color information to improve accuracy. Recent progress is mainly achieved by region-based approaches [28, 40, 41, 36]. The underlying premise is that color statistics of object regions can be distinguished from the background. With constant advancements in recent years, region-based approaches now possess the capability to track objects with efficiency and accuracy even in noisy and cluttered images, only utilizing a texture-less 3D model. However, a drawback of these optimization-based methods is the requirement for handcrafted features and carefully tuned hyper-parameters, which may not be robust in real-world scenarios.

Recently, end-to-end learning-based approaches have

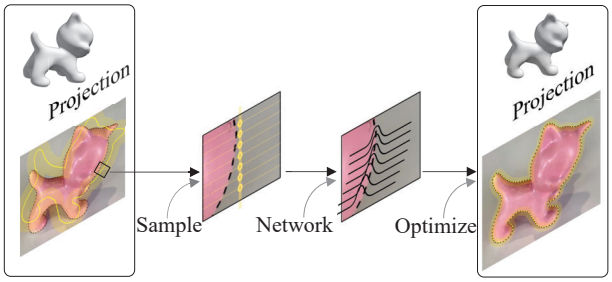

Figure 1. **The basic idea of deep active contours.** Given the initial object pose, we project the object model to the image, sample a set of points on the projected contours, use a network to predict the distribution of true boundary location along the normal at each sampled point, and optimize the pose to align the projected contours with the predicted boundary.

*The first two authors contributed equally. The authors from Zhejiang University are affiliated with the State Key Lab of CAD&CG and ZJU-SenseTime Joint Lab of 3D Vision.

†Corresponding author: Xiaowei Zhou.

been proposed to enhance the robustness of 6-DoF object detection and tracking. These approaches regress geometric parameters such as camera poses [54, 16] and object coordinates [3, 12, 45], or adopt render-and-compare [23, 15, 55, 24, 48] strategies to iteratively refine the pose. Despite the demonstrated promising results, pose regression methods exhibit limited accuracy and poor generalization, while render-and-compare approaches are computationally expensive and inapplicable for real-time applications.

In this paper, we present a learning-based active contour model (DeepAC), for real-time 6-DoF object tracking. By combining the benefits of traditional optimization-based and learning-based methods, DeepAC achieves both robustness and real-time performance. Inspired by the region-based approach RBGT [35], DeepAC takes the local region around the projected contours as input and predicts the directions to update the contours. Unlike traditional methods that rely on handcrafted features and statistical hypotheses, a network is employed to estimate the directions, as depicted in Figure1. Specifically, the proposed method presents a three-phase pipeline. First, DeepAC employs an FPN-Lite network with MobileNetV2 [31] to extract multi-level features for the current image and projects the 3D object model to acquire the 2D contours from the last frame pose. Then, a boundary prediction network is designed which utilizes the features of the local regions around the contours as input and outputs a probability distribution of the true boundary locations. Finally, the 6-DoF object pose is optimized using Newton's method based on the boundary probability. The optimization process is differentiable w.r.t. the network output, allowing the use of ground-truth poses as supervision to train the feature extraction and boundary prediction networks, hence eliminating the need for handcrafted intermediate supervision.

We validate the effectiveness of our proposed method on both semi-synthetic and real-world 6-DoF object-tracking datasets. The results demonstrate that DeepAC surpasses other optimization-based and learning-based baselines by a large margin. Moreover, we demonstrate the real-time performance of our algorithm on a mobile device, achieving a frame rate of 25fps on iPhone 11. Please see the demo video in the supplementary material.

Our key contributions are summarized below:

- A novel learning-based active contour model for real-time 6-DoF object tracking.
- A lightweight network to evolve the contours based on image features, which ensures both robustness and efficiency.
- An efficient optimization algorithm that allows the whole pipeline to be trained end-to-end with pose supervision.

2. Related Work

Keypoint-based optimization. Early keypoint-based approaches [25, 30, 19] involve establishing 2D-3D correspondences by utilizing local feature matching [34, 42, 18, 17] or optical flow [11, 43, 27] techniques. Despite demonstrating a remarkable performance, this method necessitates the presence of textured object models.

Edge-based optimization. To alleviate the need for textured models, researchers have turned to edge-based methods that commonly rely on the analysis of object edge displacement. For example, RAPiD [6] estimates the relative pose between consecutive frames by seeking pronounced gradients in close proximity to the projected edges along the orthogonal directions. To bolster tracking stability, Simon and Berger [33] implement robust estimation techniques that reduce the impact of outliers on RAPiD optimization. Further improvements include incorporating local color information [32], integrating a particle filter for initialization [44, 39], and adding edge weight for pose optimization [44, 14]. Nevertheless, edge-based methods still encounter difficulties to deal with background clutter and motion blur.

Region-based optimization. More recently, region-based methods have demonstrated remarkable success in tracking texture-less objects in complex environments. The line of research can be traced back to the work of PWP3D [28], which effectively combines color segmentation statistic models and object render boundary distance fields to optimize object pose. Subsequent work on RBOT-estimation [40] and RBOT-tracking [41] has expanded on this approach, incorporating temporally consistent local color histograms as well as utilizing the Gaussian-Newton method to optimize the energy function. RBGT [35] has introduced the precomputed sparse correspondence lines of multiple viewpoints and built a probabilistic model that conforms to a Gaussian distribution. This allows for fast convergence of object pose using the Newton method. The latest advancement in this field, SRT3D [36], has introduced smoothed step functions that account for both global and local uncertainties, providing a notable improvement over existing methods. ICG method [37] has achieved better performance than several learning-based approaches by integrating depth information with region statistics. However, the performance of region-based object tracking is often compromised by the need to define multiple handcraft features and parameters, which presents a notable limitation in real applications.

Learning-based approaches. Recent years have witnessed significant progress in deep learning methods in the

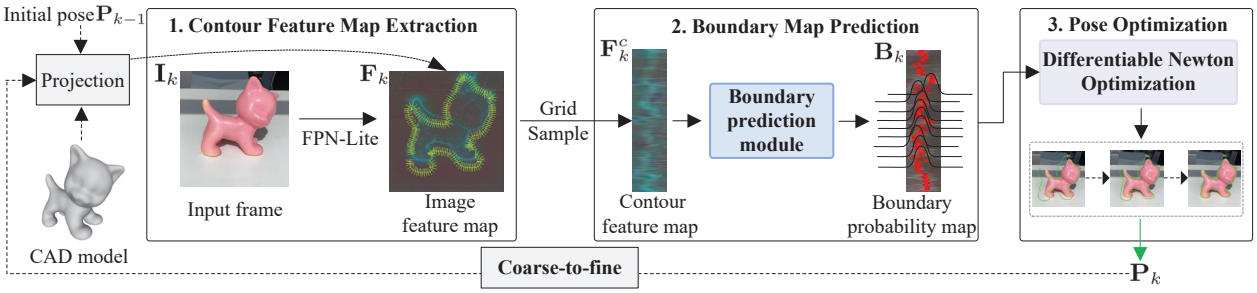

Figure 2. **Overview of the proposed method.** 1. The method uses an FPN-Lite CNN to extract multi-level features $\mathbf{F}_k$ for the current cropped frame $\mathbf{I}_k$, and represents the local region of the contours by a correspondence line model. (Section 3.2). 2. A contour feature map $\mathbf{F}_k^c$ is built by sampling a cycle of correspondence lines upon the image feature map, followed by a boundary prediction module that predicts boundary location probability $\mathbf{B}_k$ (Section 3.3). 3. A differentiable optimization layer is used to estimate the pose $\mathbf{P}_k$ in a coarse-to-fine manner (Section 3.4).

realm of 6-DoF object pose estimation. One approach involves directly predicting rotation and translation parameters, as seen in works such as [54, 16]. Another approach [3, 12, 45] generates 2D-3D correspondences by regressing object coordinates corresponding to each pixel, followed by estimating the 6-DoF pose with PnP solvers. However, accurately estimating object pose in a single-shot setting can be challenging. To overcome this, various studies [23, 15, 55, 24, 48] have utilized iterative refinement techniques that produce more precise results. The key idea behind this approach involves an iterative "render-and-compare" scheme. In each iteration, the current object pose estimate is used to render the 3D model, and the rendered image is compared to the actual image to obtain a pose update that improves the alignment between the two. PoseRBPF [4] trains a codebook to estimate the posterior probability of the particle filter for instance-level object tracking. Recent object tracking methods [47, 49, 50] have integrated depth information to remove the need for CAD models. The primary deficiency of current learning-based methods is that they require the use of a high-end GPU, making them unsuitable for deployment in mobile applications, e.g., augmented reality. Our framework, instead, addresses this limitation by incorporating a lightweight neural network into an optimization-based method, allowing for fast processing on mobile devices while achieving notable improvements in pose accuracy.

3. Methods

3.1. Overview

Given a 3D CAD model $\mathcal{M}$, an image sequence $\{\mathbf{I}_k\}$ and an initial pose in the first frame, the proposed method takes a single RGB image $\mathbf{I}_k$ of the current frame and the pose $\mathbf{P}_{k-1}$ of the previous frame to iteratively recover the current pose $\mathbf{P}_k$. First, the 3D model $\mathcal{M}$ is projected to the image plane to obtain the 2D contours using the previous pose $\mathbf{P}_{k-1}$ and extract the contour feature map according to the local region (Section 3.2). Subsequently, the contour feature map is utilized to estimate a boundary probability map (Section 3.3). Finally, the current pose $\mathbf{P}_k$ is optimized based on the boundary probability map (Section 3.4). An overview of the proposed method is provided in Figure 2.

3.2. Contour Feature Map Extraction

We use the pose $\mathbf{P}_{k-1}$ estimated from the previous frame to initialize the pose $\mathbf{P}_k$ of the current frame. The pose $\mathbf{P}$ is defined as $[\mathbf{R}, \mathbf{t}]$, where $\mathbf{R}$ is 3D rotation and $\mathbf{t}$ is 3D translation. Then we project the 3D model $\mathcal{M}$ to acquire the 2D contours and calculate its 2D bounding box, which is used to extract the cropped image $\mathbf{I}_k$. The 3D model $\mathcal{M}$ is represented by a triangle mesh with vertices $\mathbf{X}_i = [X_i, Y_i, Z_i]^\top \in \mathbb{R}^3$, where $i = 1, \ldots, n$.

An FPN-Lite network with MobileNetV2 [31] is employed to extract multi-level features from the cropped image $\mathbf{I}_k$. The resulting feature maps are denoted by $\mathbf{F}_k \in \mathbb{R}^{W_s \times H_s \times D_s}$, which represent a range of coarse-to-fine features, where s denotes the level index. The coarse-to-fine design enables the encoding of a larger spatial context in the image, which enhances tracking accuracy, particularly in cases involving large displacements. We visualize the multi-level feature maps in Figure 3. The architecture of the proposed network is detailed in the supplementary material.

Inspired by RBGT [35], we represent the local region of contours by the correspondence line model. Specifically, we sample several 2D points on the contours and build associated correspondence lines $\{\boldsymbol{l}_i\}$. A correspondence line can be represented by a center point $\mathbf{c}_i = [c_{x_i}, c_{y_i}]^\top \in \mathbb{R}^2$ and a unit normal vector $\mathbf{n}_i = [n_{x_i}, n_{y_i}]^\top \in \mathbb{R}^2$, which are obtained by projecting a 3D contour point $\mathbf{X}_{c_i}$ and its associated normal vector $\boldsymbol{N}_{c_i}$ onto the 2D image plane. A 2D point on the correspondence line is represented by $\boldsymbol{l}_i(r) = \mathbf{c}_i + r\mathbf{n}_i$. We fix the length of each

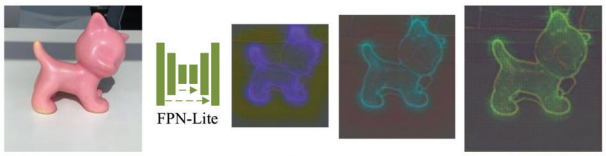

Figure 3. **Multi-level feature maps.** The feature maps of different levels reflect different receptive fields. PCA is used to reduce the dimension of the feature maps $\mathbf{F}_k^i$ which are then visualized in RGB color.

correspondence line as a constant and sample a discrete set of $\bar{r} \in \{-m, \ldots, 0, \ldots, m\}$ to generate a 2D point set $\{\boldsymbol{l}_i(\bar{r})\}$, which is used to extract the contour feature map $\mathbf{F}_k^c \in \mathbb{R}^{(2m+1)\times n_c \times D_s}$, where n_c is the number of sampled 2D contour points.

3.3. Boundary Map Prediction

We use a lightweight CNN to extract the boundary probability map $\mathbf{B}_k \in \mathbb{R}^{(2m-2w+1)\times n_c \times 1}$, where w is utilized to filter the border range. In this map, the value at location $(\bar{r}+m-w, i)$ refers to the probability of the 2D point $\boldsymbol{l}_i(\bar{r})$ being the boundary of the ith correspondence line. To enhance the generalization capacity of our network, we fuse the statistical foreground probability map $\mathbf{FG}_k^c$ and the statistical boundary probability map $\widetilde{\mathbf{B}}_k$ with the contour feature map $\mathbf{F}_k^c$ as the inputs to the network. Figure 4 illustrates the detailed procedure of forward propagation of the lightweight CNN. We compute $\mathbf{FG}_k^c$ and $\widetilde{\mathbf{B}}_k$ using an RGB histogram statistics approach based on the contour RGB map $\mathbf{I}_k^c$, which is extracted in a manner similar to the contour feature map $\mathbf{F}_k^c$. The formulations of these statistical terms are provided in detail in the supplementary material. The experiment in Section 4 demonstrates that combining statistical information and deep features leads to a significant improvement in object tracking performance.

3.4. Pose Optimization

Based on the boundary probability map $\mathbf{B}_k$, we aim to iteratively recover the 6-DoF object pose $\mathbf{P}_k$ via an optimization procedure. As the pose $\mathbf{P}_k$ is updated after each iteration, we calculate the projected difference d_i between the unmoved center $\mathbf{c}_i$ and the varied contour point $\mathbf{X}_{c_i}$ using the following equation:

$$d_i = \mathbf{n}_i^\top (\pi(\mathbf{R}_k \mathbf{X}_{c_i} + \mathbf{t}_k) - \mathbf{c}_i), \tag{1}$$

with π the pinhole camera projection function

$$\pi(\mathbf{X}) = \begin{bmatrix} \frac{X}{Z} f_x + p_x \\ \frac{Y}{Z} f_y + p_y \end{bmatrix}, \tag{2}$$

where the focal lengths are represented by f_x and f_y, and the principal point coordinates for the directions x and y, given in pixels, are represented by p_x and p_y.

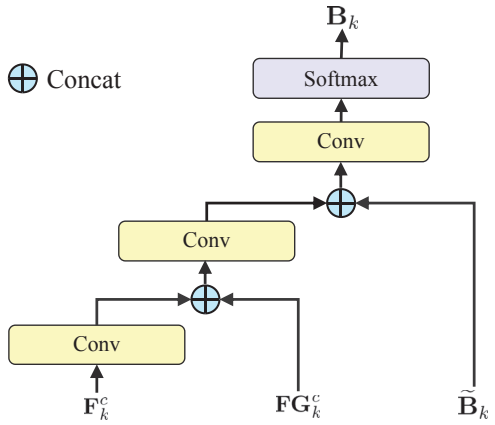

Figure 4. **Boundary prediction module.** We detail the module used for the generation of the boundary probability map $\mathbf{B}_k$, which involves the utilization of three distinct input maps, namely the contour feature map $\mathbf{F}_k^c$, the statistical foreground probability map $\mathbf{FG}_k^c$, and the statistical boundary probability map $\widetilde{\mathbf{B}}_k$. We incorporate statistical information into the deep features at multiple stages through a concatenation process.

We define the likelihood of the boundary for a correspondence line:

$$p(\boldsymbol{\mathcal{D}}_i|\mathbf{P}_k) = \mathbf{B}_k(d_i + m - w, i). \tag{3}$$

Taking all independent correspondence lines into account, the full likelihood is computed as:

$$p(\boldsymbol{\mathcal{D}}|\mathbf{P}_k) \propto \prod_{i=1}^{n_c} p(\boldsymbol{\mathcal{D}}_i|\mathbf{P}_k). \tag{4}$$

To maximize this likelihood, we adopt the iterative Newton method with Tikhonov regularization, following the approach adopted by RBGT [35]. The update rule for the pose $\mathbf{P}_k$ is as follows:

$$\begin{aligned} \Delta\boldsymbol{\theta} &= \Big(-\boldsymbol{H} + \begin{bmatrix} \lambda_r \boldsymbol{E}_3 & \mathbf{0} \\ \mathbf{0} & \lambda_t \boldsymbol{E}_3 \end{bmatrix} \Big)^{-1} \boldsymbol{g}, \\ \mathbf{P}_k &= \mathbf{P}_k \begin{bmatrix} \exp([\Delta\boldsymbol{\theta}^r]_\times) & \Delta\boldsymbol{\theta}^t \\ \mathbf{0} & 1 \end{bmatrix}, \end{aligned} \tag{5}$$

where $\boldsymbol{H}$ is the Hessian matrix, $\boldsymbol{g}$ is the gradient vector, and $\boldsymbol{E}_3$ is the 3×3 identity matrix. We use regularization parameters λ_r and λ_t for rotation and translation, respectively. Additionally, the rotation $\mathbf{R}$ is represented by the exponential map of the Lie algebra $\mathfrak{so}(3)$:

$$\mathbf{R} = \exp([\boldsymbol{\theta}^r]_\times) = \sum_{n=0}^{\infty} ([\boldsymbol{\theta}^r]_\times)^n, \tag{6}$$

where $[\boldsymbol{\theta}^r]_\times$ is the skew-symmetric matrix of $\boldsymbol{\theta}^r \in \mathbb{R}^3$ and $[\boldsymbol{\theta}^r]_\times \in \mathfrak{so}(3)$. Therefore, we also represent a pose by a

6-DoF variation $\boldsymbol{\theta}^\top = [\boldsymbol{\theta}^{r^\top}, \boldsymbol{\theta}^{t^\top}], \boldsymbol{\theta}^t \in \mathbb{R}^3$. The Hessian matrix and gradient vector are calculated using the chain rule:

$$\boldsymbol{g}^\top = \sum_{i=1}^{n_c} \frac{\partial \ln p(\boldsymbol{\mathcal{D}}_i|\boldsymbol{\theta})}{\partial d_i} \frac{\partial d_i}{\partial \mathbf{X}_{c_i}^{cam}} \frac{\partial \mathbf{X}_{c_i}^{cam}}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=0}, \tag{7}$$

$$\boldsymbol{H} \approx \sum_{i=1}^{n_c} \frac{\partial^2 \ln p(\boldsymbol{\mathcal{D}}_i|\boldsymbol{\theta})}{\partial {d_i}^2} \Big(\frac{\partial d_i}{\partial \mathbf{X}_{c_i}^{cam}} \frac{\partial \mathbf{X}_{c_i}^{cam}}{\partial \boldsymbol{\theta}}\Big)^\top \Big(\frac{\partial d_i}{\partial \mathbf{X}_{c_i}^{cam}} \frac{\partial \mathbf{X}_{c_i}^{cam}}{\partial \boldsymbol{\theta}}\Big)\Big|_{\boldsymbol{\theta}=0}, \tag{8}$$

where $\mathbf{X}_{c_i}^{cam} = \mathbf{R}_k \mathbf{X}_{c_i} + \boldsymbol{t}_k$ is in the camera frame and the derivation of $\frac{\partial d_i}{\partial \mathbf{X}_{c_i}^{cam}} \frac{\partial \mathbf{X}_{c_i}^{cam}}{\partial \boldsymbol{\theta}}$ is provided in detail in the supplementary material.

To end-to-end train our network, we employ two approximations to estimate the first-order derivative of $\ln p(\boldsymbol{\mathcal{D}}_i|\boldsymbol{\theta})$ with respect to d_i. The first approximation directly computes the derivative using the mean μ_i and variance σ_i^2 for each correspondence line, resulting in:

$$\frac{\partial \ln p(\boldsymbol{\mathcal{D}}_i|\boldsymbol{\theta})}{\partial d_i} \approx -\frac{1}{\sigma_i^2}(d_i - \mu_i). \tag{9}$$

$$\begin{aligned} \mu_i &= \sum_{\bar{r}=-(m-w)}^{m-w} \bar{r}\mathbf{B}_k(\bar{r}+m-w, i), \\ \sigma_i^2 &= \sum_{\bar{r}=-(m-w)}^{m-w} (\bar{r}-\mu_i)^2 \mathbf{B}_k(\bar{r}+m-w, i). \end{aligned} \tag{10}$$

The second approximation involves computing the first-order derivative as:

$$\frac{\partial \ln p(\boldsymbol{\mathcal{D}}_i|\boldsymbol{\theta})}{\partial d_i} \approx \frac{\alpha_s}{\sigma_i^2} \ln \frac{\mathbf{B}_k(d_i^+ + m - w, i)}{\mathbf{B}_k(d_i^- + m - w, i)}, \tag{11}$$

where d_i^+ and d_i^- are the upper and lower neighboring discrete projected differences of d_i, and α_s is the step size used in [36]. For the second-order derivative, we use the following equation:

$$\frac{\partial^2 \ln p(\boldsymbol{\mathcal{D}}_i|\boldsymbol{\theta})}{\partial {d_i}^2} \approx -\frac{1}{\sigma_i^2}. \tag{12}$$

The two approximations alternately optimize the object pose in two iterations per level and serve different functions. On the one hand, the first approximation (9) is employed to learn the boundary location μ_i of each correspondence line, allowing for fast convergence. On the other hand, the second approximation (11) is utilized to learn the local boundary probabilities for detailed refinement. In addition, our network learns the uncertainty σ_i^2 of each correspondence line, which improves robustness during optimization. In Figure 5, we visualize these uncertainties as boundary heatmaps over the images.

Figure 5. **Visualization of the boundary uncertainties.** Warmer colors indicate higher certainties, showing which parts of the object boundary are most helpful for predicting the object pose.

3.5. Supervision

We minimize the reprojection errors between the 3D points transformed by the pose estimated at each iteration $\mathbf{P}_k^{it}, it = \{1, \ldots, N_{it}\}$ and the ground truth pose $\mathbf{P}_k^{gt}$:

$$\mathcal{L} = \frac{1}{N_{it}} \sum_{it} \sum_{i} \rho(\|\pi(\mathbf{R}_k^{it}\mathbf{X}_i + \mathbf{t}_k^{it}) - \pi(\mathbf{R}_k^{gt}\mathbf{X}_i + \mathbf{t}_k^{gt})\|_2^2), \tag{13}$$

where ρ is the Huber robust kernel. To prevent hard examples from smoothing the fine features, we set a condition that the loss function is applied only when the previous iteration has succeeded in bringing the estimated pose within a satisfactory range of the ground truth. In cases where the estimated pose has not yet reached this threshold, subsequent loss terms are ignored, ensuring that only reliable training examples are used to refine the model.

3.6. Implementation Details

Our model is trained on 4 Tesla V100 GPUs for 3 hours employing the Adam optimizer with an initial learning rate of 1×10^{-3} and a batch size of 48. We utilize the FPN-Lite network with MobileNetV2 [31] as the encoder, which is initialized with pre-trained weights provided by MobileNetV2. We extract n_{level}=3 feature maps with dimensions $D_{1,2,3}$=16 and strides of 1, 2, and 4. For boundary map prediction, we employ a lightweight CNN consisting of 7 convolutional layers with randomly initialized weights. We sample n_c=200 correspondence lines on the projected contours and use m=9 to sample discrete $\bar{r}$. We set w to 4, λ_r to 5000, λ_t to 500000, α_s to 1.3 and N_{it} to 6. We train our model on six datasets from the BOP [10] challenge, namely IC-BIN [5], T-LESS [8], TUD-L [9], LM [7], YCB-V [54], and RU-APC [29]. We divide the objects into two categories: training objects and validation objects, and use their associated sequences for training and evaluation, respectively. Since not all of these datasets are sequential, we generate initial poses by adding random noises to the ground-truth poses for input images during training.

4. Experiments

4.1. Evaluation Protocols

Datasets. We evaluate our method on three standard object tracking benchmarks, namely, *RBOT* [41], *BCOT* [21] and *OPT* [52] datasets. The *RBOT* dataset comprises 18 distinct objects and features 4 sequences per object with different variations, including regular, dynamic light, noisy, and occlusion scenes. The *BCOT* dataset consists of 20 textureless objects, 22 scenes, and 404 video sequences, encompassing a total number of $126K$ frames captured in real-world environments with various camera settings, indoor/outdoor locations, and motion modes. The *OPT* dataset contains 6 objects and 552 real-world sequences with diverse lighting conditions and preset trajectories recorded by a robot arm.

Baselines. We compare the proposed method with the following baselines in two categories: 1) *Optimization-based baselines*, which encompasses keypoint-based [51, 26, 1], edge-based [22, 2, 14, 38, 44, 39] and region-based [28, 41, 56, 57, 13, 20, 35, 36] methods that share similar settings to our approach. 2) *Learning-based baselines* [23, 24], that utilize a render-and-compare framework to estimate the relative pose between the single image and a predefined textured model. We include these learning-based baselines in our analysis to show the superior generalization performance of our proposed method on unseen objects. It is important to note that our method can operate in real-time on mobile devices and solely relies on textureless objects, while [23, 24] cannot achieve this due to the extensive time overhead and the need for textured models.

Metrics. We employ various metrics, including the cm-degree score, the ADD score, and the area under curve (AUC) score, to evaluate the tracking performance.

The cm-degree score assesses the performance of the tracking algorithm by calculating the percentage of frames where the predicted pose $\mathbf{P} = [\mathbf{R}, \mathbf{t}]$ of the object has an error of less than a specified number of centimeters in translation and degrees in rotation compared to the ground-truth pose $\mathbf{P}^{gt} = [\mathbf{R}^{gt}, \mathbf{t}^{gt}]$. The definitions of the rotation error e_r and the translation error e_t are as follows:

$$\begin{aligned} e_r &= \arccos\Big(\frac{\mathrm{trace}(\mathbf{R}^\top \mathbf{R}^{gt}) - 1}{2}\Big) \\ e_t &= \|\mathbf{t} - \mathbf{t}^{gt}\|_2 \end{aligned} \tag{14}$$

For example, the $5cm$-5° score refers to the success rate of tracking poses with a translation error of less than 5 centimeters and a rotation error of less than 5 degrees.

The ADD score first measures the average distance e_v between the vertices $\mathbf{X}_i$ of a 3D model $\mathcal{M}$ transformed by the predicted pose $\mathbf{P}$=$[\mathbf{R}, \mathbf{t}]$ and the ground-truth pose $\mathbf{P}^{gt}$=$[\mathbf{R}^{gt}, \mathbf{t}^{gt}]$, as follows:

$$e_v = \frac{1}{N}\sum_{i}^{N} \|(\mathbf{R}\mathbf{X}_i + \mathbf{t}) - (\mathbf{R}^{gt}\mathbf{X}_i + \mathbf{t}^{gt})\|_2 \tag{15}$$

Then, the ADD-kd score counts the proportion of the tracking frames whose average vertices distance e_v is less than k times the diameter d of the target object.

The AUC score is another metric used to measure tracking performance. It is determined by the area under the ADD-kd curve, where the horizontal axis represents k and the vertical axis represents the ADD-kd score. For instance, the $\mathrm{AUC}(0, K)$ score is obtained by integrating the ADD-kd value within the $k \in [0, K]$ interval:

$$\mathrm{AUC}(0, K) = \int_0^K \text{ADD-}k\text{d} \cdot dk \tag{16}$$

We follow the standard tracking evaluation protocol of previous works [35, 36, 39], which states that the estimated pose should be reset to the ground truth if the rotation error exceeds 5° or the translation error exceeds $5cm$. This

Method	Regular	Dynamic Light	Noisy	Unmodeled Occlusion	Reset Times
Tjaden et al. [41]	79.9	81.2	56.6	73.3	-
Zhong et al. [57]	82.7	81.3	63.6	78.4	-
Li et al. [22]	85.8	86.7	71.4	80.3	-
Huang et al. [14]	86.9	87.3	65.0	83.6	-
Sun et al. [38]	88.1	88.8	80.5	85.1	-
Huang et al. [13]	89.9	90.7	69.6	88.9	-
RBGT [35]	90.0	90.6	71.5	85.6	-
SRT3D [36]	94.2	94.6	81.7	93.2	6575
LDT3D [39]	95.2	95.4	83.2	**94.9**	6228
DeepAC	**95.6**	**95.6**	**88.0**	94.0	**4826**

Table 1. **Comparison to optimization-based methods on the *RBOT* benchmark.** We report the tracking successful rate below the threshold of $5cm$-5° and the number of times the pose is reset.

	ADD-			cm-degree		Reset
Method	0.02d	0.05d	0.1d	5-5	2-2	Times
Wang et al. [44]	5.5	32.7	64.6	54.4	12.4	-
tjaden et al. [41]	11.7	31.6	57.1	77.1	40.8	-
Huang et al. [14]	12.0	31.3	57.5	84.1	45.1	-
Li et al. [20]	9.1	31.5	58.1	95.0	38.5	-
Huang et al. [13]	15.6	39.8	66.1	87.1	51.4	-
Li et al. [22]	14.4	38.1	65.7	87.0	50.2	-
RBGT [35]	10.9	45.5	76.9	89.0	46.0	-
SRT3D [36]	12.5	49.4	82.1	93.1	53.6	8548
LDT3D [39]	15.3	52.1	82.7	93.8	63.2	7789
DeepAC	**24.4**	**66.2**	**92.3**	**94.0**	**65.5**	**7547**

Table 2. **Comparison to optimization-based methods on the *BCOT* benchmark.** The metrics include the cm-degree scores, the ADD scores with $\{2\%, 5\%, 10\%\}$ of the object diameter, and the number of times the pose is reset.

criterion is applied to all metrics on the *RBOT* and *BCOT* datasets, but not for the *OPT* dataset. The first frame of each sequence is initialized with the ground-truth pose and the 2D bounding box is computed from the pose of the previous frame. We additionally report the number of times the pose is reset and the tracking results without any pose reset on the *RBOT* and *BCOT* datasets.

4.2. Experiment Results.

Comparison to optimization-based methods. On the *RBOT* dataset, the accuracy computation follows prior studies [41] employing the standard $5cm$-5° score. Table 1 presents a comparison between our proposed method and various counterparts. The experimental results indicate that for regular, dynamic light, and occlusion scenes, the existing optimization-based methods have reached a performance plateau, with success rates exceeding 95%. In such cases, DeepAC exhibits a similar performance. However, for noisy scenes, our method achieves significantly better performance, with an average success rate improvement from 83.2% to 88.0%. This observation demonstrates the superior robustness to noise of DeepAC.

On the *BCOT* dataset, we adopt both the ADD score and the cm-degree score to quantify pose error. As opposed to the *RBOT* benchmark, we assess tracking performance using more rigorous criteria, including ADD-0.02d, ADD-0.05d, and $2cm$-2° scores, in order to evaluate high-precision tracking capabilities. The results are presented in Table 2, which demonstrate that DeepAC exhibits superior performance compared to all baselines across all ADD and cm-degree scores. Notably, our method exhibits a remarkable advantage in terms of very strict ADD criteria, including a 9.1% improvement at ADD-0.02d, a 14.1% improvement at ADD-0.05d, and a 9.6% improvement at ADD-0.1d. These results suggest that our approach is highly effective in high-precision tracking. Besides, we select the three best-performing methods, i.e., SRT3D [36], LDT3D [39] and DeepAC to compare the number of times of pose reset, and the tracking results without any pose reset on *RBOT* and *BCOT* datasets, which are presented in Table 3. DeepAC achieves the best results on all metrics except $5cm$-5° on *BCOT* dataset and outperforms the other two baseline methods by a large margin.

On the *OPT* dataset, following [52], we present the $\mathrm{AUC}(0, 0.2)$ score as a metric for evaluating object tracking performance. Table 4 demonstrates that our method outperforms the current state-of-the-art optimization-based approaches for all six objects. These outcomes emphasize the efficacy of DeepAC in real-world applications. Furthermore, we observe that LDT3D [39], which ranks second in both *RBOT* and *BCOT*, exhibits a substantial drop in performance on the *OPT* dataset. This result may be attributed to the relatively small frame difference of the *OPT* dataset, where the non-local optimization employed in LDT3D brings a negative impact on its performance.

	ADD-			cm-degree	
Methods	0.02d	0.05d	0.1d	5-5	2-2
RBOT					
SRT3D [36]	7.0	12.3	15.9	19.0	15.1
LDT3D [39]	10.3	15.5	18.1	18.8	17.3
DeepAC	**16.6**	**26.5**	**30.3**	**30.3**	**27.8**
BCOT					
SRT3D [36]	9.2	36.2	60.7	**69.6**	41.9
LDT3D [39]	10.5	34.5	54.2	61.4	43.7
DeepAC	**17.9**	**46.8**	**64.7**	65.5	**48.0**

Table 3. **Tracking results without any pose reset on *RBOT* and *BCOT* datasets.** We report the cm-degree scores and the ADD scores of the three best-performing methods.

Method	Soda	Chest	Ironman	House	Bike	Jet	**Avg.**
PWP3D [28]	5.87	5.55	3.92	3.58	5.36	5.81	5.01
ElasticFusion [51]	1.90	1.53	1.69	2.70	1.57	1.86	1.87
UDP [1]	8.49	6.79	5.25	5.97	6.10	2.34	5.82
ORB-SLAM2 [26]	13.44	15.53	11.20	17.28	10.41	9.93	12.97
Bugaev et al. [2]	14.85	14.97	14.71	14.48	12.55	17.17	14.79
Tjaden et al. [41]	8.86	11.76	11.99	10.15	11.90	13.22	11.31
Zhong et al. [56]	9.01	12.24	11.21	13.61	12.83	15.44	12.39
Li et al. [22]	9.00	14.92	13.44	13.60	12.85	10.64	12.41
SRT3D [36]	15.64	16.30	17.41	16.36	13.02	15.64	15.73
LDT3D [39]	4.20	9.20	3.26	4.05	7.63	8.65	6.16
DeepAC	**15.71**	**17.63**	**17.58**	**18.01**	**13.91**	**17.17**	**16.67**

Table 4. **Comparison to optimization-based methods on the *OPT* benchmark.** We report the AUC scores for the range of 0% to 20% of the object diameter without any pose reset.

Comparison to learning-based methods. To demonstrate the generalization capabilities of DeepAC, we conduct a comparative experiment with two learning-based methods [23, 24] on the *RBOT* dataset, regardless of their heavy time cost. Specifically, we train the model of DeepIM [23] on the YCB-V [54] dataset using the source code from their official repository[1], while we test the pre-trained model[2] provided by [24] that is trained on the YCB-V [54] dataset. For a fair comparison, we train our DeepAC model with the same training configuration on the YCB-V [54] dataset, denoted as DeepAC$^-$. The experimental results shown in Table 5 indicate that DeepAC$^-$ outperforms the learning-based baselines [23, 24] by a significant margin, even when ground truth object masks are provided to [24], demonstrating our capability to generalize across different datasets.

[1] https://github.com/NVlabs/DeepIM-PyTorch

[2] https://github.com/princeton-vl/Coupled-Iterative-Refinement

Method	ADD-			cm-degree	
	0.02d	0.05d	0.1d	5-5	2-2
DeepIM [23]	3.9	19.1	39.2	32.9	11.6
Lipson et al. [24]	2.1	10.5	30.5	65.4	24.9
Lipson et al. [24]$^+$	10.9	36.6	69.1	**91.3**	56.7
DeepAC$^-$	**39.1**	**69.2**	**86.8**	89.9	**70.0**

Table 5. **Comparison to learning-based methods on the *RBOT* benchmark.** We present the cm-degree scores and ADD scores across all sequences, excluding the clown object sequence as it contains an incorrect texture map. [24]$^+$ means using ground-truth masks as input.

Parameters	ADD-			cm-degree	
	0.02d	0.05d	0.1d	5-5	2-2
RBOT					
no statistics	42.7	74.9	90.1	90.9	74.9
no multi-level	34.8	65.6	84.9	83.3	64.1
no uncertainty	40.4	72.7	89.4	90.8	74.5
Full	43.9	76.4	91.3	93.3	78.8
BCOT					
no statistics	22.4	61.3	87.8	87.6	57.2
no multi-level	20.5	58.0	84.1	83.1	51.7
no uncertainty	23.7	64.9	90.9	92.4	63.3
Full	24.4	66.2	92.3	94.0	65.5

Table 6. **Ablation study.** Three variants of DeepAC are trained and evaluated on the *RBOT* and *BCOT* datasets.

Ablation studies. We validate the design choices in DeepAC: 1) statistical information combination, 2) multi-level features, 3) correspondence line uncertainties, 4) the number of correspondence lines, and 5) the number of samples on each correspondence line. We conduct these experiments using the same training and evaluation protocol as 6-DoF object tracking on the *RBOT* and *BCOT* datasets. The results presented in Table 6 demonstrate that all the design choices in DeepAC bring significant performance gains. Table 7 provides results on the impact of the number of correspondence lines and samples on each line, allowing us to balance between accuracy and efficiency.

4.3. Implementation on Mobile Devices.

In addition to implementing DeepAC on desktops, we port it to mobile devices (iPhone 11). Specifically, we utilize coremltools[3] to facilitate the network deployment of DeepAC and implement the supplementary modules using C++.

Pose initialization. To initialize the pose in the first frame, we project the 3D model to the phone screen via a predefined pose and ask the user to manually move the cellphone to match the real object and the projected model. The process is illustrated in the demonstration video in the supplementary material. The initialization is deemed successful in the following way. First, we estimate the boundary positions μ_i and uncertainties σ_i^2 using our neural network. Then, we compute the average distance between the boundary positions and the midpoints of the correspondence lines, as well as the average uncertainty. If the computed distance is below a certain threshold, we use this pose as initialization and start tracking.

[3]https://github.com/apple/coremltools

Parameters	ADD-			cm-degree	
	0.02d	0.05d	0.1d	5-5	2-2
RBOT					
$(n_c, m) = (50, 9)$	32.4	64.6	84.1	85.8	59.1
$(n_c, m) = (100, 9)$	41.7	74.0	89.4	91.8	74.0
$(n_c, m) = (200, 9)$	43.9	76.4	91.3	93.3	78.8
$(n_c, m) = (300, 9)$	44.2	76.6	91.4	93.3	79.4
$(n_c, m) = (200, 5)$	38.9	66.2	83.8	84.1	62.9
$(n_c, m) = (200, 7)$	46.7	77.5	91.2	92.0	77.7
$(n_c, m) = (200, 9)$	43.9	76.4	91.3	93.3	78.8
$(n_c, m) = (200, 11)$	45.4	77.8	92.3	93.3	78.8
BCOT					
$(n_c, m) = (50, 9)$	17.3	51.8	79.8	88.1	46.7
$(n_c, m) = (100, 9)$	22.9	63.2	89.3	93.2	60.7
$(n_c, m) = (200, 9)$	24.4	66.2	92.3	94.0	65.5
$(n_c, m) = (300, 9)$	24.2	65.7	92.3	93.6	66.0
$(n_c, m) = (200, 5)$	23.0	61.6	86.1	87.8	57.7
$(n_c, m) = (200, 7)$	25.3	67.1	91.8	92.8	65.5
$(n_c, m) = (200, 9)$	24.4	66.2	92.3	94.0	65.5
$(n_c, m) = (200, 11)$	25.3	67.4	92.5	94.2	65.6

Table 7. **Sensitivity analysis** on different numbers of correspondence lines and samples on each correspondence line on the *RBOT* and *BCOT* datasets.

Running time. We analyze the time cost for each individual module in DeepAC on iPhone 11. The image preprocessing and the FPN-Lite network module take 6ms and 8.1ms, respectively. The modules for extracting correspondence lines, contour features, and boundary maps require 5.1ms, 3.7ms, and 4.2ms, respectively. The pose optimization takes 4.2ms and the color histogram updating takes 0.7ms. Overall, the whole pipeline combining all these modules achieves a running speed of approximately 25 frames per second on average.

5. Conclusion

This paper presented a learning-based active contour model, named DeepAC, for real-time 6-DoF object tracking from an RGB video. With the initial pose, the proposed DeepAC uses a three-stage pipeline to track the object: contour feature map extraction, boundary map prediction and pose optimization. The experiments showed that DeepAC achieved state-of-the-art results on multiple semi-synthetic and real-world 6-DoF object tracking datasets, surpassing both conventional optimization-based and recent learning-based methods, while being able to run in real-time on a mobile device.

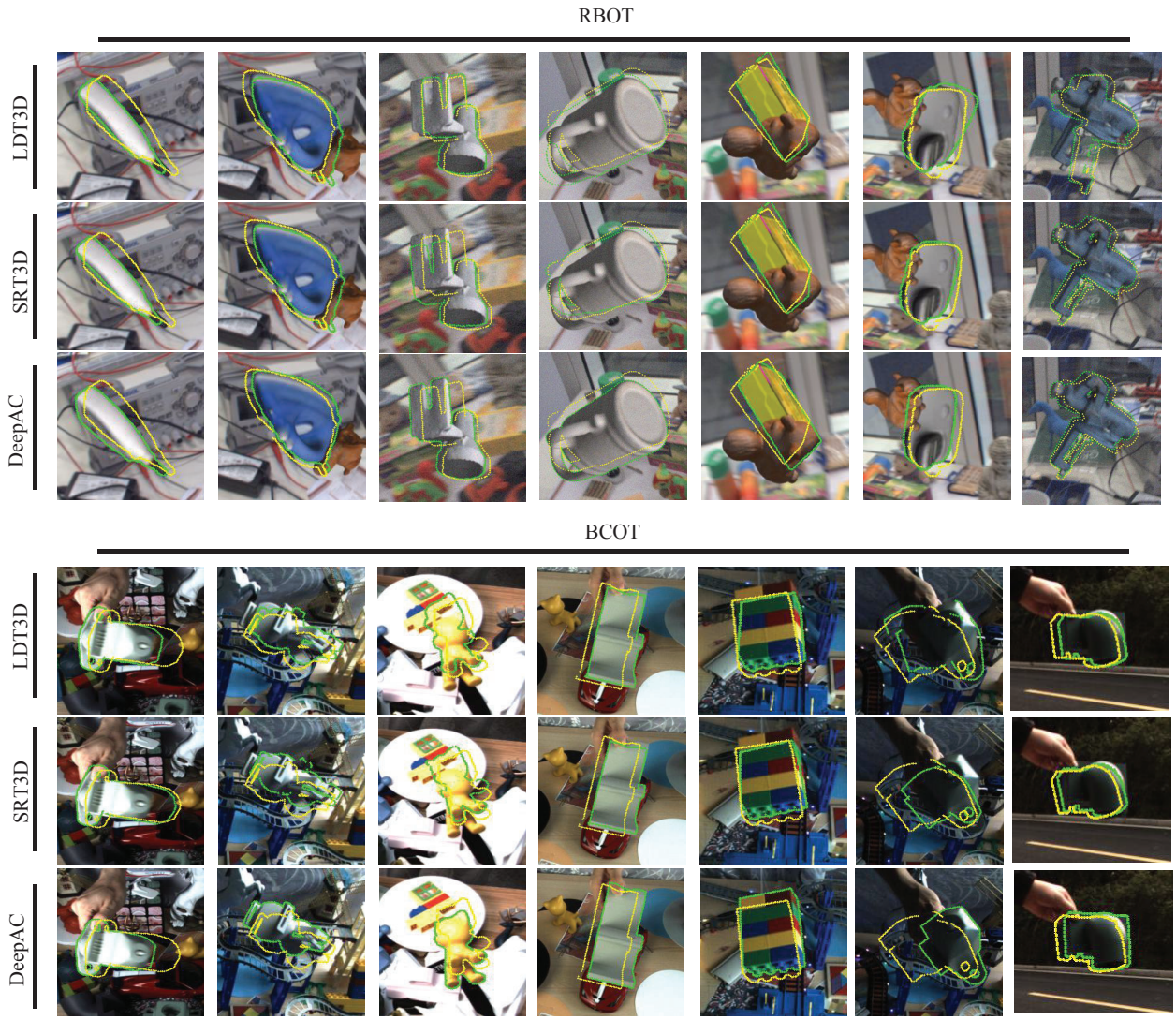

Figure 6. **Qualitative results.** We compare DeepAC to SRT3D [36] and LDT3D [39] on the *RBOT* and *BCOT* datasets. DeepAC achieves superior results in object tracking, particularly in challenging conditions involving background clutter, noise, and illumination changes. The yellow and green contours correspond to the initial pose and the optimized pose, respectively.

Acknowledgements. The authors would like to acknowledge the support from the National Key Research and Development Program of China (No. 2020AAA0108901), NSFC (No. 62171451, No. 62101576, No. 62071478), and ZJU-SenseTime Joint Lab of 3D Vision.

References

[1] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *CVPR*, 2016.

[2] Bogdan Bugaev, Anton Kryshchenko, and Roman Belov. Combining 3d model contour energy and keypoints for object tracking. In *ECCV*, 2018.

[3] Bo Chen, Alvaro Parra, Jiewei Cao, Nan Li, and Tat-Jun Chin. End-to-end learnable geometric vision by backpropagating pnp optimization. In *CVPR*, 2020.

[4] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. Poserbpf: A rao-blackwellized particle filter for 6-d object pose tracking. *IEEE Trans. Robot.*, 2021.

[5] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. Recovering 6d object pose and predicting next-best-view in the crowd. In *CVPR*, 2016.

[6] Chris Harris and Carl Stennett. RAPID - a video rate object tracker. In *BMVC*, 1990.

[7] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *ACCV*, 2013.

[8] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: an RGB-D dataset for 6d pose estimation of texture-less objects. In *WACV*, 2017.

[9] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, et al. BOP: benchmark for 6d object pose estimation. In *ECCV*, 2018.

[10] Tomáš Hodaň, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, and Jiří Matas. BOP challenge 2020 on 6d object localization. In *ECCVW*, 2020.

[11] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artif. Intell.*, 1981.

[12] Yinlin Hu, Pascal Fua, Wei Wang, and Mathieu Salzmann. Single-stage 6d object pose estimation. In *CVPR*, 2020.

[13] Hong Huang, Fan Zhong, and Xueying Qin. Pixel-wise weighted region-based 3d object tracking using contour constraints. *TVCG*, 2021.

[14] Hong Huang, Fan Zhong, Yuqing Sun, and Xueying Qin. An occlusion-aware edge-based method for monocular 3d object tracking using edge confidence. *Comput. Graph. Forum.*, 2020.

[15] Shun Iwase, Xingyu Liu, Rawal Khirodkar, Rio Yokota, and Kris M Kitani. Repose: Real-time iterative rendering and refinement for 6d object pose estimation. *arXiv:2104.00633*, 2021.

[16] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: making rgb-based 3d detection and 6d pose estimation great again. In *ICCV*, 2017.

[17] Kiyoung Kim, Vincent Lepetit, and Woontack Woo. Keyframe-based modeling and tracking of multiple 3d objects. In *ISMAR*, 2010.

[18] Alexander Ladikos, Selim Benhimane, and Nassir Navab. A real-time tracking system combining template-based and feature-based approaches. In *VISAPP*, 2007.

[19] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. BRISK: Binary robust invariant scalable keypoints. In *ICCV*, 2011.

[20] Jiachen Li, Xiuqiang Song, Fan Zhong, and Xueying Qin. Fast 3d texture-less object tracking with geometric contour and local region. *Comput. Graph.*, 2021.

[21] Jiachen Li, Bin Wang, Shiqiang Zhu, Xin Cao, Fan Zhong, Wenxuan Chen, Te Li, Jason Gu, and Xueying Qin. BCOT: A markerless high-precision 3d object tracking benchmark. In *CVPR*, 2022.

[22] Jia-Chen Li, Fan Zhong, Song-Hua Xu, and Xue-Ying Qin. 3d object tracking with adaptively weighted local bundles. *JCST*, 2021.

[23] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *ECCV*, 2018.

[24] Lahav Lipson, Zachary Teed, Ankit Goyal, and Jia Deng. Coupled iterative refinement for 6d multi-object pose estimation. In *CVPR*, 2022.

[25] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[26] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Trans. Robot.*, 2017.

[27] Youngmin Park, Vincent Lepetit, and Woontack Woo. Extended keyframe detection with stable tracking for multiple 3d object tracking. *IEEE Trans. Vis. Comput. Graph.*, 2010.

[28] Victor Adrian Prisacariu and Ian D. Reid. PWP3D: real-time segmentation and tracking of 3d objects. *IJCV*, 2012.

[29] Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robot. Autom. Lett.*, 2016.

[30] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, 2011.

[31] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[32] Byung-Kuk Seo, Hanhoon Park, Jong-Il Park, Stefan Hinterstoisser, and Slobodan Ilic. Optimal local searching for fast and robust textureless 3d object tracking in highly cluttered backgrounds. *IEEE Trans. Vis. Comput. Graph.*, 2013.

[33] Gilles Simon and Marie-Odile Berger. A two-stage robust statistical method for temporal registration from features of various type. In *ICCV*, 1998.

[34] Iryna Skrypnyk and David G Lowe. Scene modelling, recognition and tracking with invariant image features. In *ISMAR*, 2004.

[35] Manuel Stoiber, Martin Pfanne, Klaus H Strobl, Rudolph Triebel, and Alin Albu-Schäffer. A sparse gaussian approach to region-based 6dof object tracking. In *ACCV*, 2020.

[36] Manuel Stoiber, Martin Pfanne, Klaus H Strobl, Rudolph Triebel, and Alin Albu-Schäffer. SRT3D: A sparse region-based 3d object tracking approach for the real world. *IJCV*, 2022.

[37] Manuel Stoiber, Martin Sundermeyer, and Rudolph Triebel. Iterative corresponding geometry: Fusing region and depth for highly efficient 3d tracking of textureless objects. In *CVPR*, 2022.

[38] Xiaoliang Sun, Jiexin Zhou, Wenlong Zhang, Zi Wang, and Qifeng Yu. Robust monocular pose tracking of less-distinct objects based on contour-part model. *IEEE Trans. Circuits. Syst. Video. Technol.*, 2021.

[39] Xuhui Tian, Xinran Lin, Fan Zhong, and Xueying Qin. Large-displacement 3d object tracking with hybrid non-local optimization. In *ECCV*, 2022.

[40] Henning Tjaden, Ulrich Schwanecke, and Elmar Schomer. Real-time monocular pose estimation of 3d objects using temporally consistent local color histograms. In *ICCV*, 2017.

[41] Henning Tjaden, Ulrich Schwanecke, Elmar Schömer, and Daniel Cremers. A region-based gauss-newton approach to real-time monocular multiple object tracking. *T-PAMI*, 2018.

[42] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Stable real-time 3d tracking using online and offline information. *T-PAMI*, 2004.

[43] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *IEEE Trans. Vis. Comput. Graph.*, 2009.

[44] Bin Wang, Fan Zhong, and Xueying Qin. Robust edge-based 3d object tracking with direction-based pose validation. *Multim. Tools Appl.*, 2019.

[45] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. GDR-Net: geometry-guided direct regression network for monocular 6d object pose estimation. In *CVPR*, 2021.

[46] Guofeng Wang, Bin Wang, Fan Zhong, Xueying Qin, and Baoquan Chen. Global optimal searching for textureless 3d object tracking. *Vis. Comput.*, 2015.

[47] Bowen Wen and Kostas Bekris. Bundletrack: 6d pose tracking for novel objects without instance or category-level 3d models. In *IROS*, 2021.

[48] Bowen Wen, Chaitanya Mitash, Baozhang Ren, and Kostas E Bekris. se(3)-tracknet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains. In *IROS*, 2020.

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

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

[51] Thomas Whelan, Stefan Leutenegger, Renato Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense SLAM without a pose graph. In *RSS*, 2015.

[52] Po-Chen Wu, Yueh-Ying Lee, Hung-Yu Tseng, Hsuan-I Ho, Ming-Hsuan Yang, and Shao-Yi Chien. A benchmark dataset for 6dof object pose tracking. In *ISMAR*, 2017.

[53] Harald Wuest and Didier Stricker. Tracking of industrial objects by using CAD models. *JVRB*, 2007.

[54] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *RSS*, 2017.

[55] Yan Xu, Kwan-Yee Lin, Guofeng Zhang, Xiaogang Wang, and Hongsheng Li. Rnnpose: Recurrent 6-dof object pose refinement with robust correspondence field estimation and pose optimization. In *CVPR*, 2022.

[56] Leisheng Zhong and Li Zhang. A robust monocular 3d object tracking method combining statistical and photometric constraints. *IJCV*, 2019.

[57] Leisheng Zhong, Xiaolin Zhao, Yu Zhang, Shunli Zhang, and Li Zhang. Occlusion-aware region-based 3d pose tracking of objects with temporally consistent polar-based local partitioning. *TIP*, 2020.