Multimedia Tools and Applications
https://doi.org/10.1007/s11042-018-6727-5

Robust edge-based 3D object tracking with direction-based pose validation

Bin Wang[1,3] · **Fan Zhong**[1] · **Xueying Qin**[2,3]

Received: 4 May 2017 / Revised: 8 May 2018 / Accepted: 25 September 2018 /
Published online: 20 October 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract

In this paper we propose a robust edge-based approach for 3D textureless object tracking. We first introduce an edge-based pose estimation method, which minimizes the holistic distance between the projected object contour and the query image edges, without explicitly searching for 3D-2D correspondences. This method is accurate with a good initialization; however, it is sensitive to occlusion and fast motion, thus often gets lost in real environments. To improve robustness, we exploit consistency of edge direction for validating the correctness of the estimated 3D pose, and further incorporate the validation scheme for robust estimation, non-local searching and failure recovery. The robust estimation adopts point-wise validation to reduce the effect of outlier, resulting in a direction-based robust estimator. The non-local searching is based on particle filter, with the pose validation for a faithful weighting of particles, which is shown to be better than the distance-based weighting. The failure recovery is based on fast 2D detection, and estimates the recovered pose by searching for 3D-2D point correspondences, with the validation scheme to adaptively determine state transition. The effectiveness of our approach is demonstrated using comparative experiments on real image sequences with occlusions, large motions and background clutters.

Keywords 3D tracking · Pose optimization · Distance field · Particle filter

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s11042-018-6727-5) contains supplementary material, which is available to authorized users.

✉ Fan Zhong
zhongfan@sdu.edu.cn

Bin Wang
binwangsdu@gmail.com

Xueying Qin
qxy@sdu.edu.cn

1 School of Computer Science and Technology, Shandong University, Qingdao, China

2 School of Software, Shandong University, Jinan, China

3 Engineering Research Center of Digital Media Technology, Ministry of Education of China, Jinan, China

1 Introduction

3D object tracking is a fundamental task in computer vision with a variety of applications in augmented reality and robotic manipulation. 3D tracking systems are expected to recover the six degrees of freedom(6DoF) pose parameters of an object relative to the camera in unknown and dynamic environments. In this paper we focus on tracking the 6DoF pose of textureless objects over time from monocular image sequences known the 3D model of the target.

Robust keypoint detectors and descriptors [2, 24, 30] are devised in last decades, several keypoint-based 3D tracking methods [23, 27, 34] have been proposed. Although these methods achieve impressive performance for textured objects, they are not applicable for textureless objects due to the lack of reliable feature matches.

For textureless objects, edges or contours are the vital visual cue that can be detected in most situations. RAPID [11] is the first edge-based 3D tracker by projecting the sampled 3D model edge points to a 2D image and aligning the projected edge points with the image edge points. Several improvements have been proposed for better 3D-2D correspondences afterwards [7, 25, 31, 33, 36, 37]. These methods are shown to be effective in some situations. However, since the 3D-2D correspondences are searched within a limited extent, it is unavoidable to produce incorrect correspondences, especially in more complicated cases of occlusions, large inter-frame motions and background clutters. On the other hand, the local search extent is difficult to be determined, large extent may result in more incorrect correspondences, while small extent will lead to sensitivity with large inter-frame motions.

Another way to leverage edges for 3D tracking is to optimize pose transition in edge distance field [9]. The distance between the predicted object contour and the query image edge in distance field is minimized by direct optimization of the 3D pose parameters. A robust cost function is also introduced in [9], by penalizing correspondences of large distance, which is helpful for suppressing outliers. However, since only distance is considered, this method is sensitive to cluttered background. In D^2CO [16], edge direction is introduced as cost measure, based on the directional chamfer matching [22], which requires to compute a 3D distance field (by discretizing edge direction into 60 layers), and thus takes more computational cost. Although these methods can deal with occlusion and cluttered background to some extend, they perform only local search, therefore are sensitive to fast motion.

In this paper, we propose an edge-based 3D tracking approach without explicit 3D-2D correspondences, and more importantly, performs robustly in real environments. Our method exploits both distance and direction, with distance for pose estimation and direction for pose validation. The distance-based pose estimation is fast and accurate with a good initialization, the estimated pose then is validated with edge direction for better robustness. The advantage of this approach is that the estimation and validation processes are inter-independent, making the validation results more faithful than those in D^2CO [16], which jointly minimizes distance and direction costs and may bias the direction-based validation. We introduce three techniques that incorporate with the direction-based validation scheme:

- **robust estimation**: we introduce a robust pose estimation method that adaptively weights matches with point-wise consistency of edge direction, and show that it is superior to previous distance-based robust estimator [9].
- **non-local searching**: for dealing with fast motion, we introduce a particle filtering approach incorporating with the distance-based estimation and direction-based validation. We adopt the first order autoregressive state dynamics and use the direction-based validation for faithful weighting of particles.

– **failure recovery**: the key to do failure recovery is to detect the failure case, which can be achieved by the direction-based validation. Our failure recovery is based on a fast 2D detector [13], and we introduce methods to automatically generate templates and calibrate the pose.

We will show that in all of the above tasks, using our direction-based validation/weighting is superior to the default distance-based approach. Comparative experiments demonstrate that the proposed method is effective on real image sequences with occlusions, large motions and background clutters.

2 Related work

3D object tracking is an active research topic, thus the literature is particularly massive. According to the type of correspondences, 3D tracking algorithms can be categorized into 3D-3D correspondences based [1, 4], 2D-2D correspondences based [12] and 3D-2D correspondences based [10, 21]. When a 3D model of the target is available in advance, 3D-2D correspondences between 3D features of the model and 2D measurements in the image are exploited for 3D tracking. A variety of 2D measurements can be used such as fiducials, keypoints and edges, thus 3D-2D correspondences based methods are further classified as fudicial-based [18], keypoint-based [23, 27, 34] and edge-based [7, 11, 19, 25, 37]. We refer the reader to [20, 26] for more details. Here we restrict ourselves to monocular edge-based methods with a 3D model of the target available.

Following the seminal RAPID tracker [11], several methods have been proposed to improve the edge-based tracker by incorporating keypoints [6, 33]. Since keypoints and their discriminative descriptors [2, 24, 30] can be extracted and matched well for textured objects, keypoints can enhance the edge-based tracker. However, it is not suitable for textureless objects which lack of rich keypoints.

As the vital visual cue of textureless objects, edges or contours are employed by edge-based trackers. To construct 3D-2D correspondences, [7, 25] adopted a precomputed convolution kernel function of the contour orientation to find the image edge point only with an orientation similar to the projected contour orientation, not the edge point with maximum gradient in the scanline. [37] proposed multiple edge hypotheses that it attributed all the local extrema of the gradient along the scanline as potential correspondences. Multiple hypotheses prevent a wrong gradient maximum from being assigned as a correspondence, but increase the computation cost. [31, 36] exploited the local region knowledge of foreground and background, and the affinity of adjacent image edge points to search the optimal correspondences. These improvements are impressive, however the robustness decreases when ambiguities between different edges occur in the scene. All of these methods assume that edge correspondences are determined by a local search in a limit extent based on a prior pose. If the prior pose sufficiently deviates from the ground truth, tracking probably fails especially when large inter-frame motions occur.

To ensure a good prior pose, multiple pose hypotheses are proposed to propagate the pose using particle filters framework. Since Isard et al. [17] applied particle filters to 2D edge tracking, various edge-based 3D tracking methods have been implemented using a particle filter framework. [19] tracked complex 3D objects by utilizing the GPU to calculate edges and evaluate pose likelihoods. [6] employed keypoint correspondences for particle initialization, and then refined the estimated pose by aligning the projected model edges and the image edges using 3D-2D correspondences explicitly. Our approach is similar to [5] by both

using the edge distance field. [5] is a tracking-by-detection framework that uses distance field for chamfer matching between offline 2D edge templates and the scene image, and the coarse pose of the matched template is used for initializing particles. It employed a standard edge-based tracking to establish the 3D-2D correspondences and predict the final pose, while our approach directly optimizes the 6DoF pose in edge distance filed. These methods can achieve impressive results especially for large inter-frame motions. However, the computation cost increases linearly with the number of particles without parallel computing techniques.

All of aforementioned methods do not take pose recovery or reinitialization into account, especially when pose tracking drifts if the object is occluded or it leaves the camera's field of view. Object detectors can be exploited to recover tracking from failures or reinitialize tracking after drifting. DOT [14] considers only image gradients to detect objects, while the performance degenerates in case of strong background clutter. Instead of considering only dominant orientations in DOT, LINE2D [13] exploits all gradient orientations in local image neighborhoods. Moreover, LINE2D proposes a novel similar measurement to prevent problems due to too strong gradients in the background. LINEMOD [13] extends LINE2D by taking 3D surface normal orientations into account if a depth sensor is available. Park et al. [28] use DOT to detect objects in the input color image and compute a coarse pose. The coarse pose is then refined using ICP-based algorithm between the depth map of matched template and the depth map of current image. Hinterstoisser et al. [15] exploit color histogram matching and ICP-based refinement [9] to estimate accurate 6DoF pose based on LINEMOD. The pose recovery solution of our method is similar to [15, 28]. These two methods are designed to manipulate RGBD data, thus they can refine the coarse pose by ICP-based method to align the depth map between the matched template and current image. Our pose recovery solution exploits only color image to track the 6DoF pose of the target. It is a more challenging and difficult problem.

3 Edge-based 3D pose estimation

In this section we introduce a basic edge-based pose estimation method. Given an image, the optimal 3D object pose is estimated by minimizing the edge matching error of projected model contours and the image edges. Pose optimization is achieved by iteratively minimizing the matching error in edge distance field. Our method is closely related with the approach in [9], which aims to register 2D/3D point sets, while we need to handle 3D to 2D registrations.

3.1 Camera model and pose parameterization

3D tracking aims to estimate the 6DoF pose of an object relative to the camera given the camera intrinsic matrix $\boldsymbol{K} \in \mathbb{R}^{3\times3}$, the image $\boldsymbol{I} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$, the 3D surface model(triangle mesh) $\boldsymbol{M}$. A 3D model point $\boldsymbol{X} \in \mathbb{R}^3$ is projected to an image pixel position $\boldsymbol{x} \in \mathbb{R}^2$ using the standard pinhole camera model as follows:

$$\bar{\boldsymbol{x}} = \boldsymbol{K} \cdot [\boldsymbol{R}(\boldsymbol{r})|\boldsymbol{t}] \cdot \bar{\boldsymbol{X}} \tag{1}$$

where $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{X}}$ are respectively the homogenous representation of $\boldsymbol{x}$ and $\boldsymbol{X}$. $\boldsymbol{t} \in \mathbb{R}^3$ and $\boldsymbol{R}(\boldsymbol{r}) : \mathbb{R}^3 \rightarrow \mathbb{R}^{3\times3}$ are respectively the translation vector and rotation matrix parameterized

by the Rodrigues rotation vector $\boldsymbol{r}$. The Rodrigues's formula $\boldsymbol{R}(\boldsymbol{r})$ is defined as the exponential map by

$$\boldsymbol{R}(\boldsymbol{r}) = \exp(\boldsymbol{r}^{\wedge}) = \cos\theta \boldsymbol{E} + (1 - \cos\theta)\boldsymbol{\alpha}\boldsymbol{\alpha}^{\top} + \sin\theta\boldsymbol{\alpha}^{\wedge} \tag{2}$$

where the angle $\theta = |\boldsymbol{r}|$, $\boldsymbol{\alpha} = \boldsymbol{r}/\theta$ and $\boldsymbol{E} \in \mathbb{R}^{3\times3}$ is an identity matrix. The skew symmetric matrix operator $^{\wedge} : \mathbb{R}^3 \rightarrow \mathbb{R}^{3\times3}$ is defined as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}^{\wedge} = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}. \tag{3}$$

In this paper the 6DoF pose is parameterized by $\boldsymbol{p} = (\boldsymbol{r}, \boldsymbol{t}) \in \mathbb{R}^6$.

3.2 Pose estimation in distance field

We formulate the pose optimization as a contour matching process by fitting the 3D contour points set $\boldsymbol{\Phi}$ to the edge distance field $\boldsymbol{D} : \mathbb{R}^2 \rightarrow \mathbb{R}$ of the image $\boldsymbol{I}$. The overview of 3D contour matching is illustrated in Fig. 1.

To generate the edge distance field $\boldsymbol{D}$, we use the canny edge detector [3] to extract the edge map, then apply a fast distance transform [8] to the edge map. For each image pixel $\boldsymbol{x}$, $\boldsymbol{D}(\boldsymbol{x})$ indicates its distance to the nearest image edge point, and encodes the approximate gradient direction towards the edges. Figure 1a, b, and c illustrate the procedure of generating edge distance field.

Given a initial pose $\boldsymbol{p_i}$ and the 3D model $\boldsymbol{M}$, we can render the depth map and extract the 2D contour pixels on it. Then 3D contour points $\boldsymbol{\Phi}$ can be easily obtained by back-projecting them to the surface of 3D model $\boldsymbol{M}$. Figure 1f, g, h, and i illustrate the procedure of sampling 3D contour points $\boldsymbol{\Phi}$. We will temporally assume that $\boldsymbol{\Phi}$ is dependent on only the initial pose $\boldsymbol{p_i}$, an outer loop iteration will be introduced later to deal with the update of $\boldsymbol{\Phi}$ with respect to the object pose.

For a 3D contour point $\boldsymbol{X_i} \in \boldsymbol{\Phi}$, the matching cost e_i is denoted as follows:

$$e_i = \boldsymbol{D}(\pi(\boldsymbol{K} \cdot [\boldsymbol{R}(\boldsymbol{r})|\boldsymbol{t}] \cdot \bar{\boldsymbol{X}}_i)) \tag{4}$$

where π transforms the homogenous coordinates into its nonhomogenous representation. Therefore, the whole matching cost E between $\boldsymbol{\Phi}$ and $\boldsymbol{D}$ is defined by following objective energy function:

$$E(\boldsymbol{r}, \boldsymbol{t}) = \sum_{\boldsymbol{X_i} \in \boldsymbol{\Phi}} e_i^2. \tag{5}$$

Starting from $\boldsymbol{p_i}$, the optimal pose $\boldsymbol{p_o}$ is calculated by iteratively minimizing (5) using Levenberg-Marquardt (L-M) algorithm:

$$\boldsymbol{p_o} = \arg\min_{\boldsymbol{r},\boldsymbol{t}} E(\boldsymbol{r}, \boldsymbol{t}). \tag{6}$$

Figure 1e shows the evolution of projected contour points from $\boldsymbol{p_i}$ to $\boldsymbol{p_o}$ by (6). Figure 1j shows the optimal pose $\boldsymbol{p_o}$ by overlying green mask on the target.

We can differentiate (5) with respect to the pose parameters $\boldsymbol{p}$ to get the Jacobian required by L-M:

$$\boldsymbol{J} = \sum_i \frac{\partial e_i}{\partial \boldsymbol{x_i}} \cdot \frac{\partial \boldsymbol{x_i}}{\partial \boldsymbol{p}} \tag{7}$$

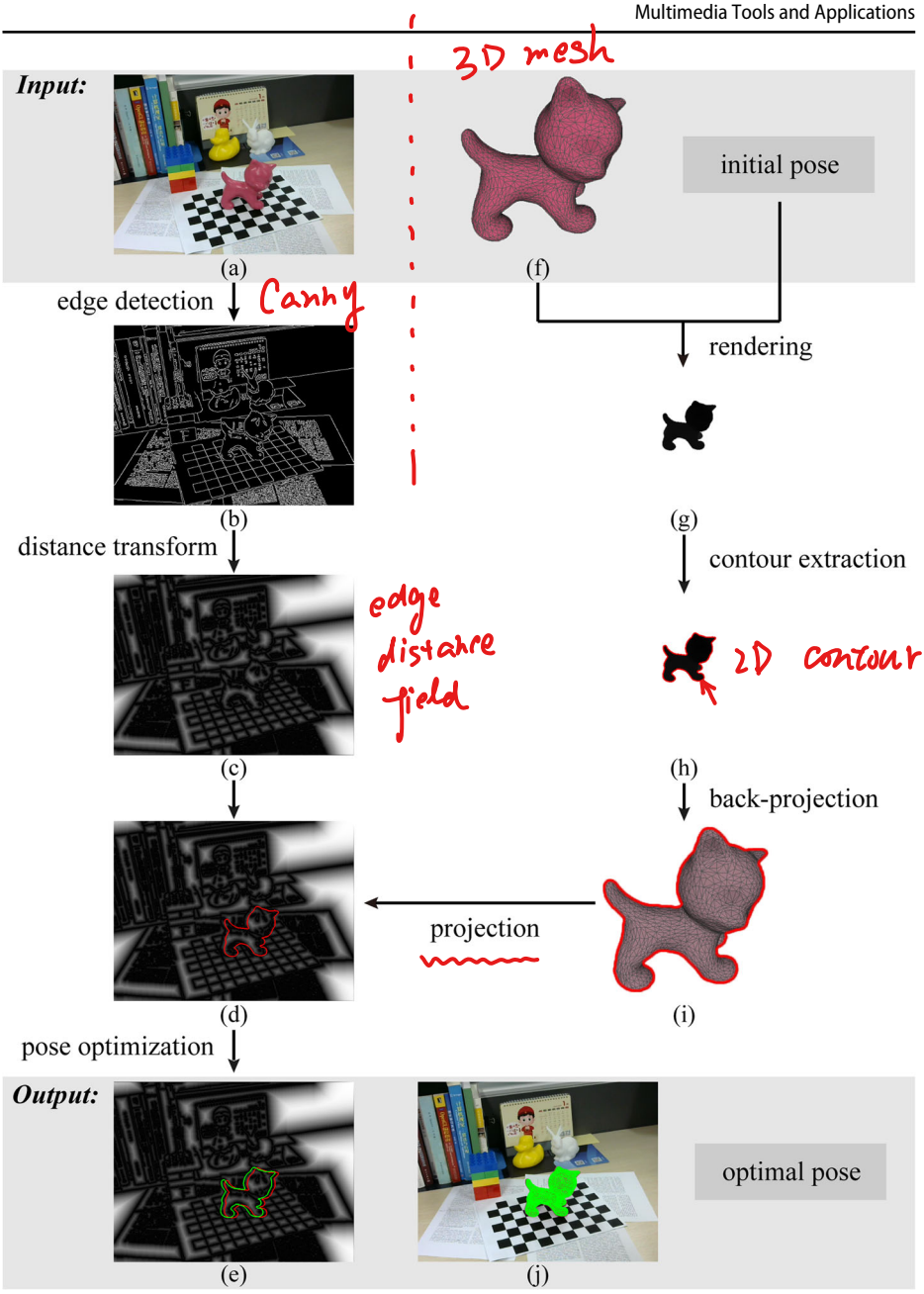


Fig. 1 Overview of 3D contour matching. **a** A color image $\boldsymbol{I}$ of the target(a pink CAT). **b** Canny edge map of $\boldsymbol{I}$. **c** Edge distance field of $\boldsymbol{I}$. **d** Projected 3D contour (in red) in edge distance field. **e** Projected 3D contour points under an initial pose (in red) to the optimal pose (in green) in edge distance field. **f** 3D mesh model of the target. **g** Depth map of 3D model under the initial pose. **h** 2D contour points (in red) of the target. **i** 3D contour points (in red) on the surface of model. **j** The optimal pose overlaid on $\boldsymbol{I}$

where $\boldsymbol{x_i}$ is the projected image pixel of $\boldsymbol{X_i}$. The differential $\frac{\partial e}{\partial \boldsymbol{x_i}} \in \mathbb{R}^{1\times 2}$ can be computed using centered finite difference in $\boldsymbol{D}$, and $\frac{\partial \boldsymbol{x_i}}{\partial \boldsymbol{p}} \in \mathbb{R}^{2\times 6}$ can be derived from (1) and (2) analytically.

Note that $\mathbf{\Phi}$ is dependent on the object pose $\boldsymbol{p}$, it should be re-sampled after the object pose updated in each L-M iteration. Actually in consideration of efficiency we did not update $\mathbf{\Phi}$ with $\boldsymbol{p}$ for each L-M iteration. It is time consuming to update $\mathbf{\Phi}$ after each L-M iteration since a graphics rendering process should be invoked. Instead we introduce an outer loop for the L-M (and the IRLS in Section 4.1) iterations, and let $\mathbf{\Phi}$ updated only at the beginning of each outer iteration, as described in Algorithm 1 (lines 10-11). For inner L-M iterations, we just update the object pose $\boldsymbol{p}$ iteratively with the 3D object contour points $\mathbf{\Phi}$ unchanged. Since $\boldsymbol{p}$ changes only a little in each L-M iteration, $\mathbf{\Phi}$ would not change much. We found that the outer iterations can converge quickly in 2 or 3 iterations, because for adjacent frames the object contour does not change much. In all of our experiments we set the outer iteration number L to 2 for better efficiency.

4 Robust tracking with pose validation

The above basic pose estimation method is accurate, provided with a good initialization and edge distance field. However, in real environments it is sensitive to occlusion, fast motion and background clutter, etc. This is largely due to the inherent local searching of L-M, and the non-weighted quadratic energy in (5). In [9], robust estimator is introduced, which can partially deal with outlier correspondences caused by occlusion and edge detection errors.

The key to improve tracking robustness is to validate the estimated pose, which in previous works is mainly achieved with the residual error of energy function (5). For example, in [9], both the Lorentzian and the Huber robust estimators are actually a soft thresholding to the matching error. We argue that this is not a good choice for edge-based tracking because, on one hand, the energy function is restricted by the optimization technique, thus not all useful features can be considered (including all non-differentiable features); on the other hand, since the residual error is minimized in a local range, it is unfaithful to be used again for validation, especially in cluttered background.

For a more faithful validation of the estimated pose, we propose to exploit the consistency of edge direction. For a 3D contour point $\boldsymbol{X_i} \in \mathbf{\Phi}$, the consistency score s_i is defined as follows:

$$s_i = |\cos(ori(\boldsymbol{x_i}) - ori(\boldsymbol{x_i'}))| \tag{8}$$

where $\boldsymbol{x_i}$ is the projected point of $\boldsymbol{X_i}$ under the estimated pose, $\boldsymbol{x_i'}$ is the closest image edge pixel of $\boldsymbol{x_i}$, $ori(\boldsymbol{x_i})$ is the normal orientation in radians of $\boldsymbol{x_i}$ and $ori(\boldsymbol{x_i'})$ is the normal orientation at $\boldsymbol{x_i'}$. Thus the direction consistency score of a pose could be defined as the average consistency of all 3D contour points by

$$S(\boldsymbol{r}, \boldsymbol{t}) = (\sum_{\boldsymbol{X}_i \in \mathbf{\Phi}} s_i)/|\mathbf{\Phi}| \tag{9}$$

where $|\mathbf{\Phi}|$ denotes the number of 3D contour points. Theoretically, the consistency score will be 1 if the estimated pose is equal to the real pose.

Note that $s_i \in [0, 1]$ is naturally a weighting function, for good matches it would be close to 1 and otherwise close to 0. It is also robust to outlier matches (occlusion, edge missing, etc.), which is less likely to keep consistency in edge directions. Unlike the case of using distance for scoring [9], the computation of $S(\boldsymbol{r}, \boldsymbol{t})$ does not involve any thresold parameter, making it more robust in real environments. We will demonstrate this with experiments (Fig. 6).

In the remaining of this section, we will incorporate the above direction-based pose validation scheme with our tracking approach, for improving the robustness. We find

three places for the validation scheme to take effect, i.e. robust pose estimation, non-local searching, and failure recovery.

4.1 Robust pose estimation

We assume that the occluded or mismatched 3D contour point tends to have a lower direction consistency score. A simple quadratic error in (5) is sensitive to these outliers. In order to supress these outliers, a weight function w should be injected into the objective function by generalizing (5):

$$E(\boldsymbol{r}, \boldsymbol{t}) = \sum_i w_i e_i^2. \tag{10}$$

how?

We can still apply the L-M algorithm to minimize the (10) simply by solving an iterated re-weighted least-squares(IRLS). In this paper we take the direction consistency score s_i as the weight function w_i.

The form of (10) is consistent with traditional distance-based robust estimator, which minimizes the following energy function:

$$E(\boldsymbol{r}, \boldsymbol{t}) = \sum_{\boldsymbol{X}_i \in \boldsymbol{\Phi}} \rho(e_i) \tag{11}$$

with $\rho(.)$ the robust estimator to penalize large error. We can differentiate (11) around the pose parameters by the chain rule:

$$\boldsymbol{J} \equiv \frac{\partial E}{\partial \boldsymbol{p}} = \sum_i \frac{\partial \rho}{\partial e_i} \cdot \frac{\partial e_i}{\partial \boldsymbol{x_i}} \cdot \frac{\partial \boldsymbol{x_i}}{\partial \boldsymbol{p}}. \tag{12}$$

So minimizing (11) requires to solve the IRLS problem of (10) with $w_i = \frac{\partial \rho}{\partial e_i} \cdot \frac{1}{e_i}$, and the difference to our method is only on the choice of w_i. As mentioned above, setting $w_i = s_i$ can take advantage the direction-based validation for better robustness.

4.2 Non-local searching with particle filtering

Generally 3D tracking starts from a prior pose $\boldsymbol{p}_t$ at frame t. Many edge-based tracking methods [7, 25, 31, 36] take the estimated pose $\boldsymbol{p}_{t-1}$ directly from the previous frame as the initial pose $\boldsymbol{p}_t$ under small inter-frame motion assumption. If large inter-frame motions occur, these methods with single pose hypothesis fail inevitably as the initial pose is not close to the global minimum. In this paper we exploit a particle filtering framework with a first order autoregressive dynamical model to deal with large inter-frame motions. Figure 2a, and b give 2 consecutive frames with a relative large inter-frame motion. Figure 2c shows the optimized pose with our particle filtering method in contrast to the estimated pose only using a single prior pose from previous frame as in Fig. 2d.

In our particle filtering framework, the posterior distribution(denoted $+$) at frame $t-1$ is represented as a set of N particles $\mathbb{S}_{t-1}^+$ associated with normalized weights Θ_{t-1}^+ by

$$\begin{cases} \mathbb{S}_{t-1}^+ = \{\boldsymbol{p}_{t-1}^{(0)}, \ldots, \boldsymbol{p}_{t-1}^{(N-1)}\} \\ \Theta_{t-1}^+ = \{\theta_{t-1}^{(0)}, \ldots, \theta_{t-1}^{(N-1)}\} \end{cases} \tag{13}$$

where the particle $\boldsymbol{p}_{t-1}^{(k)}$ is the kth sample in the 6DoF pose space with an associated weight $\theta_{t-1}^{(k)}$. For the next frame t, particles $\mathbb{S}_t^-$ are resampled according to the weights Θ_{t-1}^+ and transited by a motion model to form the prior distribution(denoted $-$) of frame t:

$$\begin{cases} \mathbb{S}_t^- = \{\boldsymbol{p}_t^{(0)}, \ldots, \boldsymbol{p}_t^{(N-1)}\} \\ \Theta_t^- = \{1/N, \ldots, 1/N\} \end{cases} \tag{14}$$

where Θ_t^- indicates each particle with an uniform weight. $\mathbb{S}_t^-$ is updated to $\mathbb{S}_t^+$ using contour-based observation as described in Section 3.2 and Θ_t^+ is evaluated according to the contour consistency score. The posterior distribution is propagated through time, and the pose of the target object is determined by taking the particle with the highest weight or the weighted mean of all particles at each frame. We describe the four steps: **initialization**, **transition**, **updating** and **resampling** in the proposed particle filtering process in more details.

Initialization At the first frame, we replicate the given starting pose to initialize $\mathbb{S}_0^-$ with uniform weights Θ_0^-.

Transition The dynamical model for particle evolution has a significant influence on tracking performance. Due to its flexibility and simplicity, we employ the first-order autoregressive dynamics for state transition, and then perturb the particle using a Gaussian noise model. For each particle $\boldsymbol{p}_t^{(k)} \in \mathbb{S}_t^-$ at frame t, the transition is processed as:

$$\boldsymbol{p}_t^{(k)} = \boldsymbol{p}_{t-1}^{(k)} + \lambda_v \boldsymbol{v}_{t-1}^{(k)} + \lambda_n \boldsymbol{n}_t^{(k)} \tag{15}$$

$$\boldsymbol{v}_{t-1}^{(k)} = \boldsymbol{p}_{t-1}^{(k)} - \boldsymbol{p}_{t-2}^{(k)} \tag{16}$$

where $\boldsymbol{v}_{t-1}^{(k)}$ denotes the velocity of the kth particle between $\boldsymbol{p}_{t-1}^{(k)} \in \mathbb{S}_{t-1}^+$ and $\boldsymbol{p}_{t-2}^{(k)} \in \mathbb{S}_{t-2}^+$. $\boldsymbol{n}_t^{(k)} \in \mathbb{R}^6$ is a Gaussian noise from $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ with a zero mean and a covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{6\times 6}$. λ_v and λ_n are the weights for balancing the autoregressive motion and random motion respectively.

Updating Once each particle at frame t is transited, it is employed in (10) as the initial pose. The updating from $\boldsymbol{p}_t^{(k)} \in \mathbb{S}_t^-$ to $\boldsymbol{p}_t^{(k)} \in \mathbb{S}_t^+$ is accomplished by optimizing the (10) iteratively:

$$\boldsymbol{p}_t^{(k)} \in \mathbb{S}_t^+ = \arg\min_{\boldsymbol{r},\boldsymbol{t}} E(\boldsymbol{r}, \boldsymbol{t}). \tag{17}$$

Figure 2e, and f illustrate the particles updated from $\mathbb{S}_t^-$ to $\mathbb{S}_t^+$, and Fig. 2g shows the evolution of the best particle in contrast to the evolution using a single prior pose as given in Fig. 2h. After the optimization is done, the estimated pose is validated by (9). Then the corresponding weight $\theta_t^{(k)}$ of $\boldsymbol{p}_t^{(k)} \in \mathbb{S}_t^+$ is evaluated using the consistency score $S(\boldsymbol{p}_t^{(k)})$ as follows:

$$\theta_t^{(k)} = S(\boldsymbol{p}_t^{(k)}). \tag{18}$$

After updating all the particles, the weight $\theta_t^{(k)} \in \Theta_t^+$ of each particle $\boldsymbol{p}_t^{(k)} \in \mathbb{S}_t^+$ is normalized by

$$\theta_t^{(k)} = \frac{\theta_t^{(k)}}{\sum_{i=1}^N \theta_t^{(i)}}. \tag{19}$$

We consider the particle $\boldsymbol{p}_t^{(k)} \in \mathbb{S}_t^+$ with the highest weight as the optimal pose at frame t.

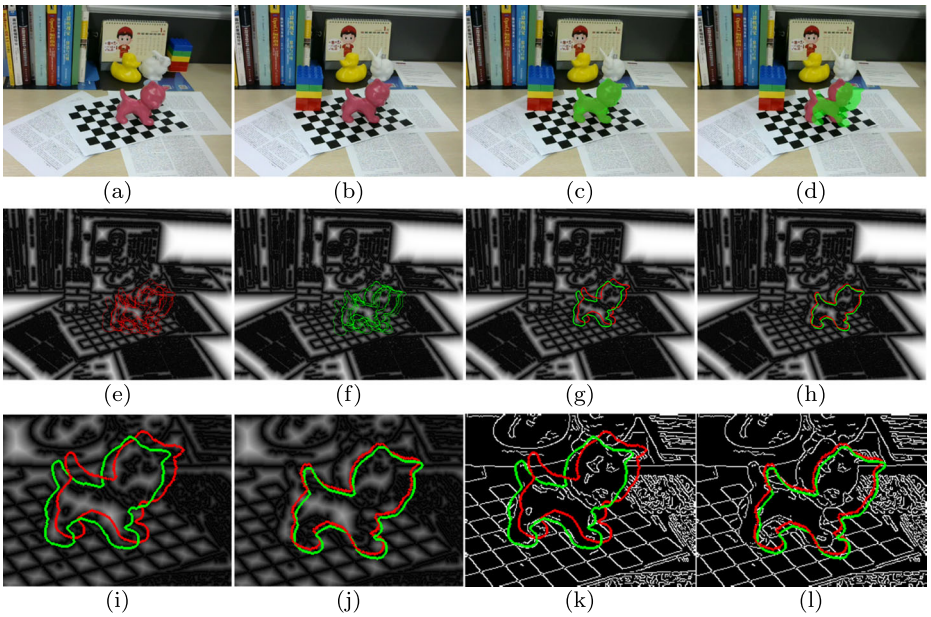


Fig. 2 Particle filtering between 2 consecutive frames of the target(a pink CAT). We take the number of particles(N) as 10 for example. **a** Frame $\boldsymbol{I}_{t-1}$ with pose $\boldsymbol{p}_{t-1}$. **b** Frame $\boldsymbol{I}_t$ with a relative large motion from $\boldsymbol{I}_{t-1}$. **c** The estimated optimal pose using our particle filtering method is visualized by green mask overlaid on the target. **d** The estimated pose with $\boldsymbol{p}_{t-1}$ as the single prior pose is visualized, obviously it converged to a local minimum. **e** The prior particles $\mathbb{S}_t^-$ sampled from $\mathbb{S}_{t-1}^+$ are visualized by red projected contours. **f** The updated particles $\mathbb{S}_t^+$ by (17) are visualized by green projected contours. **g** Projected 3D contour of the best particle is evolved using our particle filtering method from a prior pose(in red) to the optimal pose(in green) in distance field of $\boldsymbol{I}_t$. **h** Projected 3D contour is evolved from the single prior pose $\boldsymbol{p}_{t-1}$(in red) to the optimal pose(in green) in distance field of $\boldsymbol{I}_t$, it gets stuck in a local minimum. **i** The details of particles of (**g**). **j** The details of particles of (**h**). **k** The projected contour of best particle is evolved from a prior pose(in red) to the optimal pose(in green) in edge map. **l** The projected contour is evolved from the single prior pose $\boldsymbol{p}_{t-1}$(in red) to the optimal pose(in green) in edge map

Resampling When the updating is done, we obtain the posterior distribution at frame t, and it is used to generate the prior distribution at next frame $t+1$ by importance resampling. Each particle $\boldsymbol{p}_{t+1}^{(k)}$ in the prior particles $\mathbb{S}_{t+1}^-$ are randomly drawn from $\mathbb{S}_t^+$ according to the weights Θ_t^+. Therefore, a particle with higher weight is more likely propagated to the next frame in the random sampling. After resampling is done, we can return to transition step and then start the next particle filtering process.

4.3 Failure recovery

The aforementioned manipulations are designed for frame-to-frame pose tracking and do not take failure recovery into account, especially when pose tracking drifts in case of strong occlusion or when object moves out of the camera's field of view. In this section we provide boosted LINE2D for pose recovery after pose tracking is lost.

Although it is straightforward to invoke LINE2D for pose recovery, how to faithfully detect the failure cases is unaddressed and very important for the recovery procedure to

Fig. 3 Templates generation. **a** The viewpoints for rendering template images: each gray triangular pyramid represents a viewpoint. **b** A synthesized RGB color image. **c** An alpha map of the color image which is used as the foreground mask. **d** The selected features: feature points are displayed in blue, the corresponding quantized orientations are displayed in red. **e** A color image of the object in real scene, its texture could be different from the 3D model

work. Fortunately, the introduced direction-based pose validation scheme provides a simple yet effective way for failure detection. Specifically, for each frame we get an optimal pose $\boldsymbol{p_o}$ after particle filtering under (10). Then we compute the direction consistency score $S(\boldsymbol{p_o})$ under the (9). If $S(\boldsymbol{p_o}) < \lambda_d$, we consider the tracking to be lost, then a recovery process is triggered.

We synthesize template images by rendering the 3D model from positions in the upper hemisphere over the object, as shown in Fig. 3. The viewpoints are defined by subdividing a regular icosahedron recursively. A viewpoint can give us two out-of-plane rotation angles with respect to the origin of coordinate system of the object. We also consider in-plane rotations at each viewpoint. Furthermore, we adopt multiple scales by changing the size of icosahedron. In our experiments we sample 225 viewpoints, 5 in-plane rotation angles from -30° to 30° and 3 scales from 50cm to 80cm. Therefore, total 3375 template images for each object are exploited in pose recovery. LINE2D outputs a bounding box and the matched template (as shown in Fig. 4b, and c). The matched template is labeled with a 6DoF pose, as annotated during templates generation. However, since the coordinates of templates are not

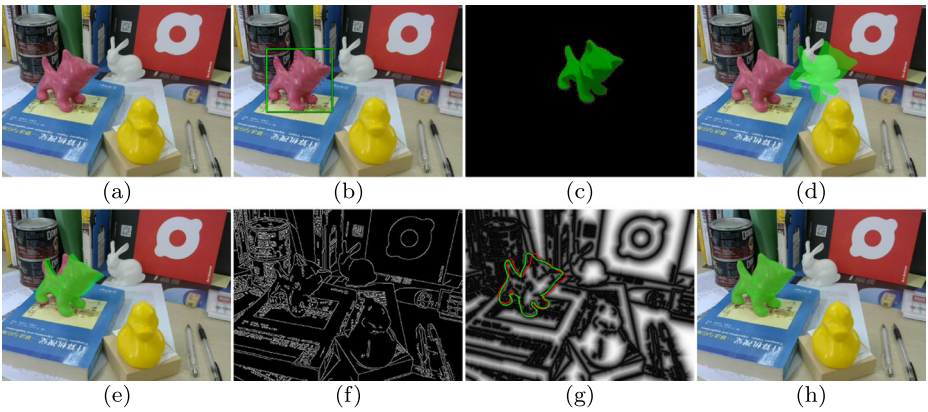


Fig. 4 Object detection and recovered pose estimation. **a** A color image of the target (a pink cat). **b** The result of 2D object detection: the bounding box is shown in green rectangle, the matched points are displayed in blue. **c** The matched template image. **d** The pose of the matched template is visualized by the green mask overlaid on the input color image. **e** The initial pose of the target calculated from (23) is visualized by the green mask. **f** The edge map of input color image. **g** Projected 3D contour points are evolved under (10) from the initial pose (in red) to the recovered pose (in green) in edge distance field. **h** The recovered pose is visualized by the green mask

aligned with the tracked object, we can't directly use the template pose for re-initialization, as shown in Fig. 4b–d.

To compute the aligned pose for a template, note that template image is aligned with the detected bounding box $\boldsymbol{b} = (x_b, y_b, w_b, h_b)$ in the current frame, that is:

$$\begin{cases} x_g = x_f + x_b \\ y_g = y_f + y_b \end{cases} \tag{20}$$

with $\boldsymbol{f} = (x_f, y_f)$ a template point and $\boldsymbol{g} = (x_g, y_g)$ the corresponding point in the current frame. Because the 3D model and the pose of matched template are available, we can obtain the 3D correspondence point $\boldsymbol{G} = (x_G, y_G, z_G)$ of $\boldsymbol{g}$ by back-projecting $\boldsymbol{f}$ to the 3D model under the pose of matched template. Therefore, we can build the correspondence between $\boldsymbol{G}$ and $\boldsymbol{g}$ via $\boldsymbol{f}$.

For a pair of point correspondence $(\boldsymbol{G}_i, \boldsymbol{g}_i)$, the distance between the projection of $\boldsymbol{G}_i$ and $\boldsymbol{g}_i$ is calculated as

$$d_i = ||\pi(\boldsymbol{K} \cdot [\boldsymbol{R}(\boldsymbol{r})|\boldsymbol{t}] \cdot \bar{\boldsymbol{G}}_i) - \boldsymbol{g}_i||_2. \tag{21}$$

Then the sum of the reprojection errors between the projection of all 3D model points and its corresponding 2D pixel points are denoted by

$$d(\boldsymbol{r}, \boldsymbol{t}) = \sum_i d_i^2. \tag{22}$$

We can estimate the initial pose $\boldsymbol{p}_d$ by minimizing (22) as

$$\boldsymbol{p}_d = \arg \min_{\boldsymbol{r},\boldsymbol{t}} d(\boldsymbol{r}, \boldsymbol{t}). \tag{23}$$

It is a classic Perspective-n-Point (PnP) problem [21] and can be solved by standard gradient-based non-linear optimization methods.

5 Implementation details

This section elaborates the complete framework of our approach and the details of parameter settings. Given an image sequence $\mathcal{I}$, the camera intrinsic matrix $\boldsymbol{K}$ and an initial pose $\boldsymbol{p}_i$, our 3D tracking approach estimates the pose $\boldsymbol{p}_t$ at each frame t with the 3D model $\boldsymbol{M}$ of target available. Finally, we get the 6DoF pose sequence $\mathcal{P}$ which contains the estimated pose parameters for each frame. The framework of our approach is summarized into Algorithm 1. The $\boldsymbol{p}_t$ is parameterized by the rotation vector $\boldsymbol{r}_t$ and translation vector $\boldsymbol{t}_t$. To minimize (10), the L-M algorithm is employed and terminated until an maximum iteration steps(100).

The number of particles N is set as a different value from {1, 10, 100} so as to evaluate the efficiency and effectiveness of particle filtering.

For simplicity, we consider elements of pose parameters are mutually independent. Thus the covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{6\times6}$ in (15) is diagonal, and $diag(\boldsymbol{\Sigma}) = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1)$.

In (15), λ_v is 0.1, and λ_n is 1. Both are used to control the transition velocity of particles. λ_d for failure detection is set as 0.8 in this paper.

Algorithm 1 Robust edge-based 3D tracking

Input: $\mathcal{I} = \{\boldsymbol{I}_0, \boldsymbol{I}_1, \cdots, \boldsymbol{I}_{T-1}\}$, $\boldsymbol{M}$, $\boldsymbol{K}$, $\boldsymbol{p}_i$
Param: N, L, $\boldsymbol{\Sigma}$, λ_v, λ_n, λ_d
Output: $\mathcal{P} = \{\boldsymbol{p}_0, \boldsymbol{p}_1, \cdots, \boldsymbol{p}_{T-1}\}$

for $t \leftarrow 0$ ***to*** $T-1$ **do**
  $EdgeMap_t \leftarrow \texttt{Canny}(\boldsymbol{I}_t)$
  $\boldsymbol{D}_t \leftarrow \texttt{DistanceTransform}(EdgeMap_t)$
  if $t = 0$ **then**
    $\mathbb{S}_t^- \leftarrow \texttt{InitializeParticles}(N, \boldsymbol{p}_i)$
    $\Theta_t^- \leftarrow \texttt{InitializeWeightsUniform}(N)$
  $\mathbb{S}_t^- \leftarrow \texttt{TransitParticles}(\mathbb{S}_t^-, \lambda_v, \lambda_n, \boldsymbol{\Sigma})$
  for $k \leftarrow 0$ ***to*** $N-1$ **do**
    $\boldsymbol{p}_t^{'} \leftarrow \boldsymbol{p}_t^{(k)} \in \mathbb{S}_t^-$
    for $j \leftarrow 1$ ***to*** L **do**
      $\boldsymbol{\Phi}_k \leftarrow \texttt{Extract3DContour}(\boldsymbol{K}, \boldsymbol{p}_t^{'}, \boldsymbol{M})$
      $\boldsymbol{p}_t^{'} \leftarrow \texttt{IRLS}(\boldsymbol{D}_t, \boldsymbol{\Phi}_k)$
    $\boldsymbol{p}_t^{(k)} \in \mathbb{S}_t^+ \leftarrow \boldsymbol{p}_t^{'}$
    $S(\boldsymbol{p}_t^{(k)}) \leftarrow \texttt{ComputeDirectionConsistency}(\boldsymbol{K}, \boldsymbol{p}_t^{(k)}, \boldsymbol{\Phi}_k, \boldsymbol{D}_t)$
    $\theta_t^{(k)} \in \Theta_t^+ \leftarrow \texttt{EvaluateWeights}(S(\boldsymbol{p}_t^{(k)}))$
  $\Theta_t^+ \leftarrow \texttt{NormalizeWeights}(\Theta_t^+)$
  $\boldsymbol{p}_t \in \mathcal{P} \leftarrow \texttt{SelectBestParticle}(\mathbb{S}_t^+, \Theta_t^+)$
  if $S(\boldsymbol{p}_t) > \lambda_d$ **then**
    $\boldsymbol{p}_t \in \mathcal{P} \leftarrow \texttt{RecoverPose}(\boldsymbol{I}_t)$
    $\mathbb{S}_{t+1}^- \leftarrow \texttt{InitializeParticles}(N, \boldsymbol{p}_t)$
    $\Theta_{t+1}^- \leftarrow \texttt{InitializeWeightsUniform}(N)$
  else
    $\mathbb{S}_{t+1}^- \leftarrow \texttt{ResampleParticles}(\mathbb{S}_t^+, \Theta_t^+, N)$
return $\mathcal{P}$

6 Experiments

In this section, we validate the proposed method using different comparative experiments. Firstly, we give a global overview of the effectiveness of our method. Then we demonstrate the effectiveness of direction-based robust estimation and non-local searching with particle filtering. Thirdly, we compare our method with a pixel-wise tracker PWP3D [29], a state-of-the-art tracker GOS [36] and D^{2}CO [16]. Finally, we adopt a marker-based tracker [18] as the baseline(the ground truth pose estimated from fiducial marker), and compare our method with it to accomplish quantitative evaluations.

Our system is implemented in C++ without parallel computing techniques, and runs on an Intel i5 CPU with 8GB RAM. The test sequences are captured by a web camera with 640 × 480 resolution, and both the object and the camera are movable. The triangle mesh models for the BUNNY, CAT, DUCK and LEGO are acquired in advance.

6.1 A global overview of tracking effectiveness

In order to give a fast global overview of our method, we demonstrate the proposed method in a sequence captured in case of scale change, rotation, fast motion and occlusion etc. Fig. 5 shows the tracking results in all above cases.

6.2 Effectiveness of direction-based robust estimation

We use a robust pose estimation method that adaptively weights matching errors with point-wise consistency of edge direction, thus the projected contour points of 3D object contour are aligned with image edge points which have similar orientation with them. Figure 6 shows it is superior to previous distance-based robust estimation [35]. Our method works fine, while the projected contour points of 3D object contour are aligned with image edge points whose orientation are not consistent with projected contour points by distance-based robust

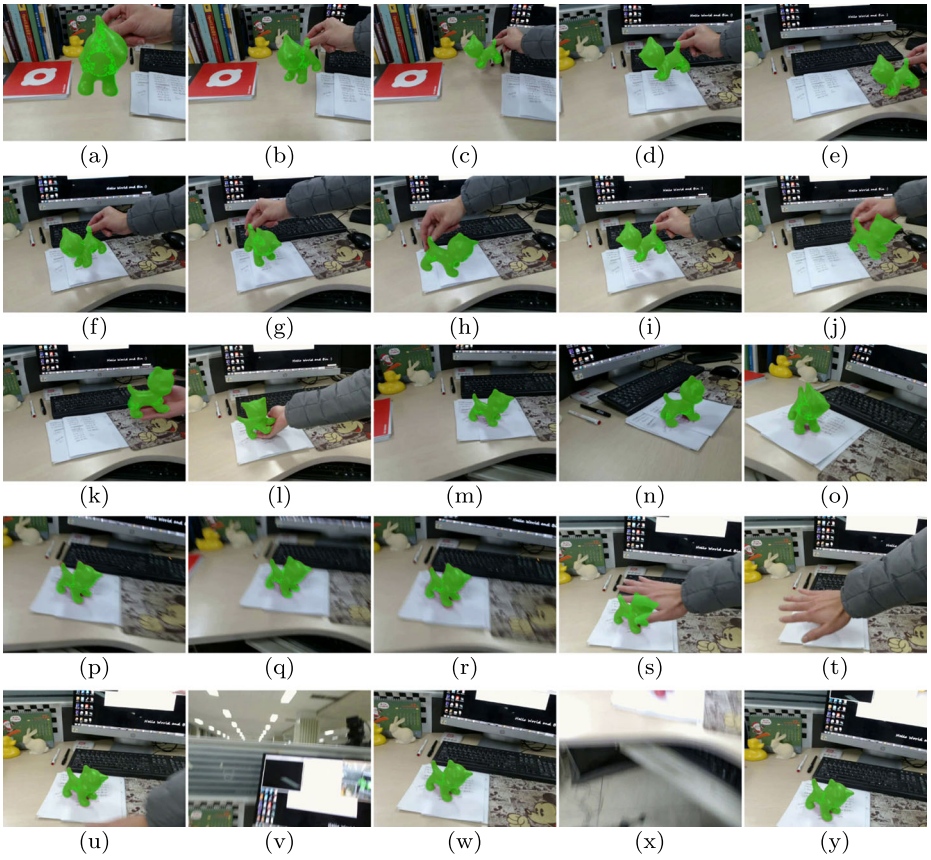


Fig. 5 An overview of tracking effectiveness with 10 particles, the tracking results are shown as a semitransparent green mask overlaid on the object. **a-c** Scale change. **d-f** Background change from simple to cluttered. **g-i** Object rotation. **j-l** Object moves out of camera's field of view, then moves in. **m-o** Camera rotation. **p-r** Strong camera shaking. **s-u** Failure recovery from occlusion. **v-y** Failure recovery in case of object out of view

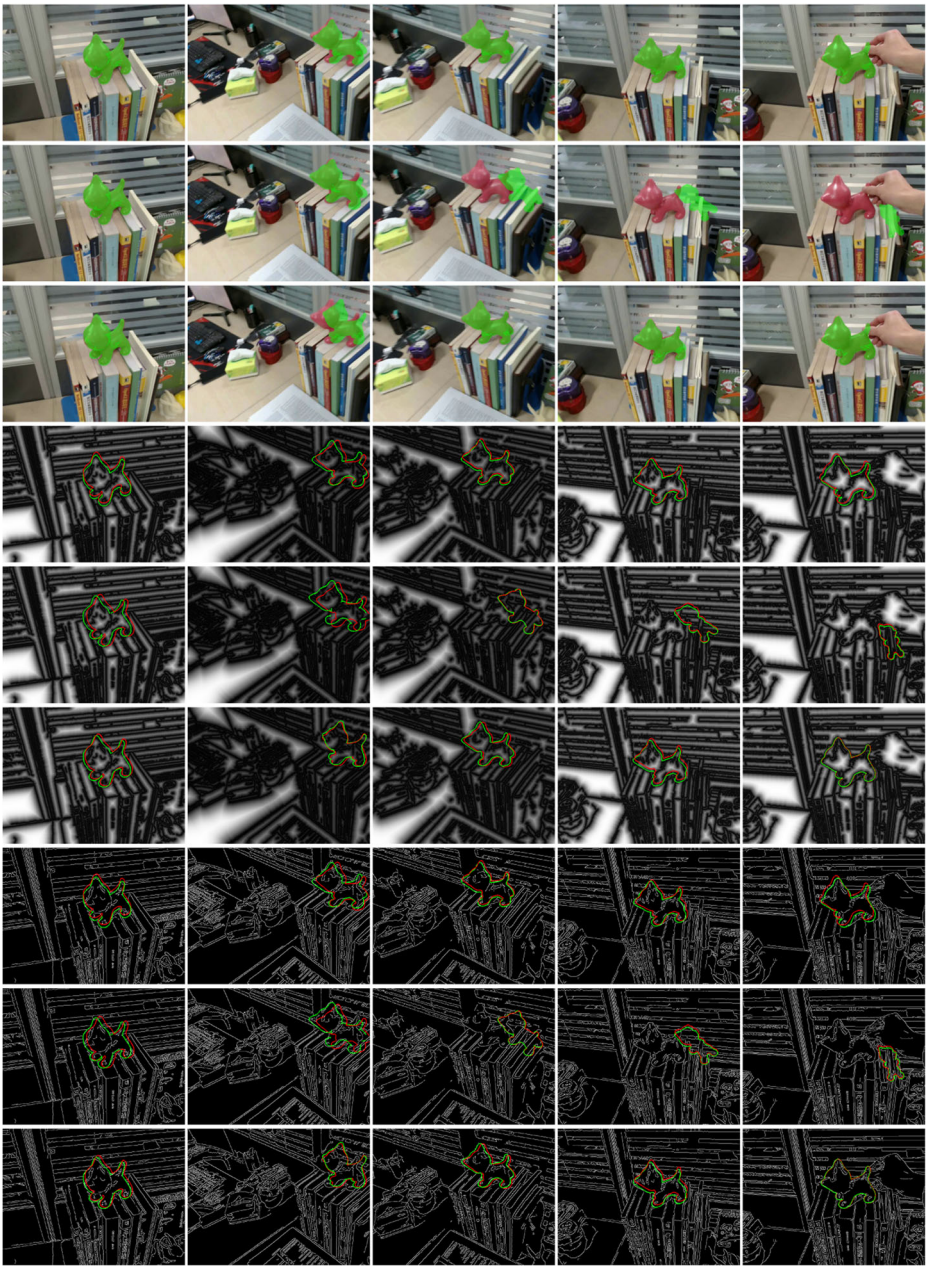


Fig. 6 Comparison between direction-based robust estimation with distance-based robust estimation using 10 particles. The first row shows the results of direction-based robust estimation in contrast to the results of distance-based robust estimation in the second row. The fourth and fifth rows show the projected contour of best particle from initial pose(in red) to estimated pose(in green) in edge distance field. The seventh and eighth rows show the projected contour of best particle from initial pose(in red) to estimated pose(in green) in edge map. The third, sixth and last rows are the corresponding results of direction-based robust estimation without particle filtering

Fig. 7 Tracking results showing the effectiveness of particle filtering in case of fast object and camera motion. The rows are respectively the tracking results with 1, 10, 100 particles. Our method with 100 particles works better than others

estimation. Figure 6 also shows that only the direction-based robust estimation cannot result into robust tracking in comparison with direction-based robust estimation using particle filtering, so we combine them and compare with distance-based method.

6.3 Effectiveness of particle filtering

Many edge-based methods with a single prior pose assume that the motion of camera or object is small and smooth, thus the prior pose is close to the global minimum. If large inter-frame motions occur, the tracking fails and the estimated pose converges to a local minimum. In order to deal with large inter-frame motions, we employ 1 particle, 10 and 100 particles to track the object under fast camera and object motion. Figure 7 gives the comparison results of 1 particle, 10 and 100 particles in case of fast camera and object motion, we could use 100 particles to track the DUCK while others drifted.

6.4 Effectiveness of failure recovery

Most 3D tracking methods do not take failure recovery into account. Our method takes advantage of LINE2D detector to recover pose tracking in case of strong occlusions or when object leaves the camera's field of view, as shown in Fig. 8, the pose of BUNNY is recovered from tracking failures by means depicted in Section 4.3.

6.5 Comparison with PWP3D tracker

We compare our method with a pixel-wised tracker PWP3D [29].[1] PWP3D proposed a probabilistic framework for simultaneous 2D image segmentation and 3D object tracking without building 3D-2D correspondences explicitly. It employed the statistic color information of foreground and background based on a prior pose, thus the tracking drifts when the target object has similar color statistics with the environment, or large inter-frame motions

[1] PWP3D: http://www.robots.ox.ac.uk/~victor/code.html

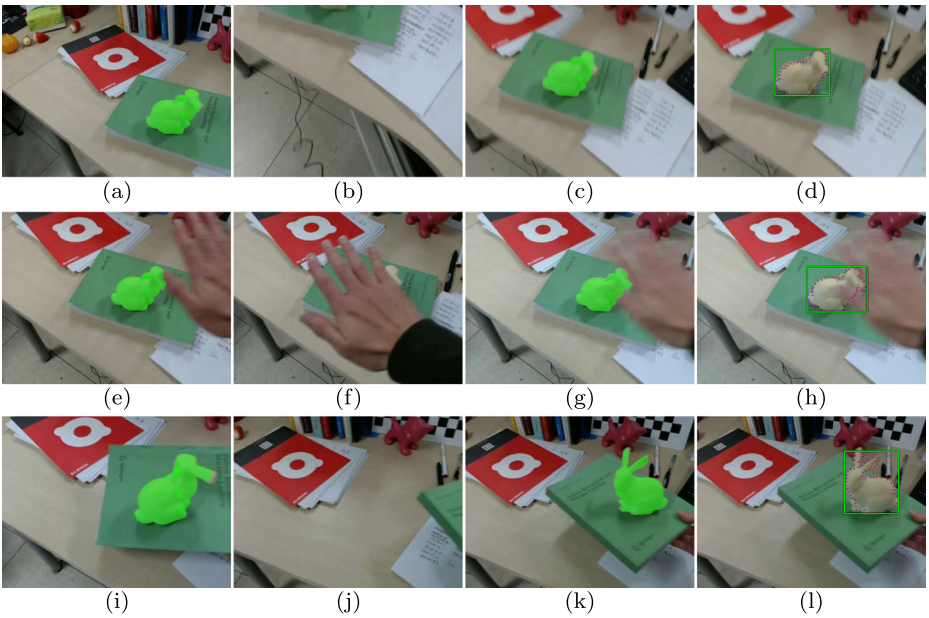


Fig. 8 Tracking results of BUNNY showing the effectiveness of failure recovery(with 10 particles). **a-c** Pose recovery from strong camera shaking. **d** The 2D detection result when the object appears after camera shaking. **e-g** Pose recovery from strong occlusion. **h** The 2D detection result when the object appears after strong occlusion. **i-k** Pose recovery after the object moves out of the camera's field of view. **h** The 2D detection result when the object move in the camera's field of view

occur. Figure 9 compares the tracking results obtained by PWP3D(the second row) with our method($N = 10$, the first row). PWP3D drifted due to the fast camera motion and white background.

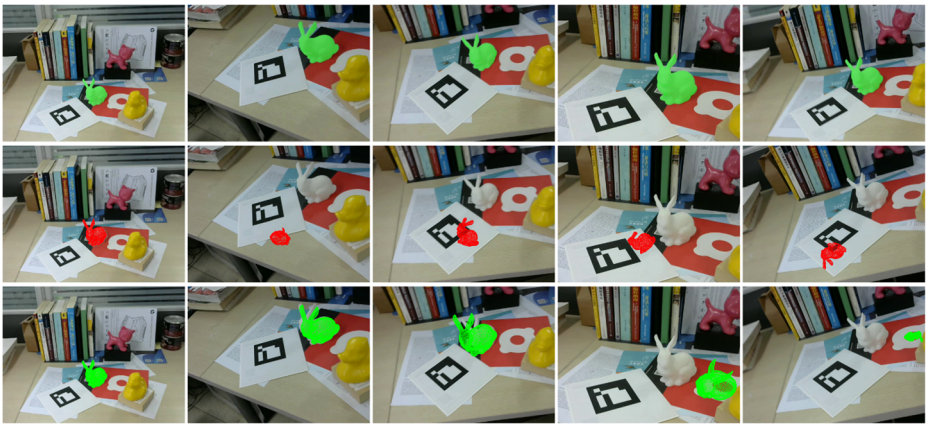


Fig. 9 Comparison between our method(with 10 particles) with PWP3D and GOS. The first row is the tracking results of our method with 10 particles. The second row is the results of PWP3D, and the tracking drifted due to fast camera motion and white background. The third row is the results of GOS, and the edge correspondences of white BUNNY are disturbed by the white edges from background

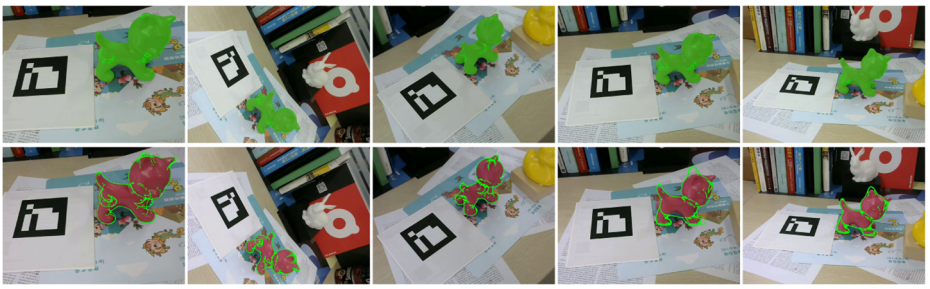


Fig. 10 Comparison between our method(with 10 particles) with D^{2}CO. The first row is the tracking results of our method with 10 particles. The second row is the results of D^{2}CO. D^{2}CO drifted due to the fast camera motion

6.6 Comparison with GOS tracker

As a representative edge-based tracker, GOS [36][2] constructs 3D-2D edge correspondences explicitly. The image edge correspondences are determined by a 1D local search with a limited extent based on a prior pose. Although GOS exploits the region knowledge around the edge point and the affinity of adjacent image edge points, it still suffers from the error correspondences raised from the similar edge of background. Figure 9 compares the tracking results between GOS(the third row) and our method($N = 10$, the first row). For the GOS tracker, the edge correspondences of white BUNNY are disturbed by the white edges from background.

6.7 Comparison with D^{2}CO tracker

Figure 10 gives the comparison between D^{2}CO [16][3] and our method with 10 particles. Our method works fine while D^{2}CO drifts.[4] In addition, we found that the results of D^{2}CO jitter significant,[5] maybe due to the discretization of edge directions, which results in a non-smooth energy function. D^{2}CO [16] jointly minimizes distance and direction costs, as mentioned in the introduction, although this is helpful for improving edge correspondence, it introduces more computations (see Table 1 for time comparison) and may bias the direction-based validation. In our approach, distance and direction are used separately for pose estimation and validation, which is more efficient, and more robust by incorporating with particle filtering.

6.8 Quantitative evaluation

In order to evaluate the tracking accuracy and time performance of our method, we adopt the marker-based tracking method [18] as the baseline. The coordinate system of the object is predefined and fixed relative to the coordinate systems of the square marker, thus we can get the ground truth pose of the object by simply transforming the pose of the marker. We use two criteria to measure the accuracy: (i) the rotation error(R) and translation error(T) [32]:

[2]GOS: https://github.com/guofengw/GOSTracker

[3]D^{2}CO: https://bitbucket.org/alberto_pretto/d2co.git

[4]For fair comparison, we closed failure recovery for comparisons with PWP3D, GOS, and D^{2}CO.

[5]See the accompany videos.

Table 1 Performance evaluation on 4 sequences. R for rotation error, T for translation error, and AD for average distance

Sequence(#)	Method	Time	Accuracy		
		(ms/frame)	R(°)	T(cm)	AD(cm)
BUNNY (1540)	PWP3D [29]	50.0	213.7	32.1	35.4
	GOS [36]	116.5	18.8	14179.0	14180.0
	D^2CO [16]	1810.8	37.3	5022.9	5022.4
	Distance-based [35]($N = 10$)	115.1	4.4	2.2	2.2
	Distance-based [35]($N = 100$)	1171.9	3.5	2.1	2.1
	Direction-based($N = 10$)	121.2	4.0	1.6	1.6
	Direction-based($N = 100$)	1232.9	3.5	1.5	1.5
CAT (1212)	PWP3D [29]	75.9	241.3	11.6	13.4
	GOS [36]	116.3	207.1	62286.4	62286.0
	D^2CO [16]	1575.9	15.0	2.7	2.1
	Distance-based [35]($N = 10$)	130.6	3.9	2.5	2.5
	Distance-based [35]($N = 100$)	1257.1	3.6	2.4	2.4
	Direction-based($N = 10$)	137.8	3.8	2.3	2.3
	Direction-based($N = 100$)	1329.4	3.6	2.2	2.2
DUCK (1677)	PWP3D [29]	73.5	189.2	7.8	9.1
	GOS [36]	143.8	75.8	3.2	3.5
	D^2CO [16]	1360.8	85.1	1107.7	1109.0
	Distance-based [35]($N = 10$)	102.5	8.0	1.7	1.7
	Distance-based [35]($N = 100$)	961.7	9.9	1.8	1.8
	Direction-based($N = 10$)	107.6	7.1	1.4	1.4
	Direction-based($N = 100$)	1012.1	6.3	1.3	1.3
LEGO (1466)	PWP3D [29]	52.3	235.8	16.9	22.6
	GOS [36]	147.6	16.2	3.5	3.3
	D^2CO [16]	1727.5	41.8	8184.7	8174.9
	Distance-based [35]($N = 10$)	86.9	3.9	1.5	0.9
	Distance-based [35]($N = 100$)	827.3	3.6	1.6	1.0
	Direction-based($N = 10$)	90.0	4.5	1.5	0.9
	Direction-based($N = 100$)	851.3	3.7	1.6	1.0
AVG (1473)	PWP3D [29]	**62.9**	220.0	17.1	20.1
	GOS [36]	131.1	79.5	19118.0	19118.2
	D^2CO [16]	1618.8	44.8	3579.5	3577.1
	Distance-based [35]($N = 10$)	108.8	5.1	2.0	1.8
	Distance-based [35]($N = 100$)	1054.5	5.2	2.0	1.8
	Direction-based($N = 10$)	114.2	4.9	**1.7**	1.6
	Direction-based($N = 100$)	1106.4	**4.3**	**1.7**	**1.5**

The bold symbols gives the best results in corresponding performance evaluations

the L2 norms of the rotation and translation components between the estimated pose and the ground truth pose are computed as follows:

$$\begin{cases} R = \frac{1}{T} \cdot \sum_{i=0}^{T-1} \left\| \hat{\boldsymbol{r}}_i - \boldsymbol{r}_i \right\|_2 \\ T = \frac{1}{T} \cdot \sum_{i=0}^{T-1} \left\| \hat{\boldsymbol{t}}_i - \boldsymbol{t}_i \right\|_2 \end{cases} \tag{24}$$

, (ii) and the average distance(AD) [15]: the average distance between all vertices of the 3D model in the estimated pose and the ground truth pose is calculated by

$$AD = \frac{1}{TV} \cdot \sum_{i=0}^{T-1} \sum_{k=0}^{V-1} \left\| (\boldsymbol{R}(\hat{\boldsymbol{r}}_i) \cdot \boldsymbol{X}_k + \hat{\boldsymbol{t}}_i) - (\boldsymbol{R}(\boldsymbol{r}_i) \cdot \boldsymbol{X}_k + \boldsymbol{t}_i) \right\|_2 \tag{25}$$

where the number of frames of a sequence is T, the total number of vertices in a 3D model is V, the estimated pose of frame i is denoted as $\hat{\boldsymbol{p}}_i = (\hat{\boldsymbol{r}}_i, \hat{\boldsymbol{t}}_i)$, and the ground truth pose of frame i is $\boldsymbol{p}_i = (\boldsymbol{r}_i, \boldsymbol{t}_i)$. For speed, we use frames per seccond(fps) as the performance metric.

The target objects are chosen with diversity in structure(symmetric vs. asymmetric). We captured 4 sequences respectively for the BUNNY(asymmetric), CAT(asymmetric), DUCK(asymmetric) and LEGO(symmetric) with a hand-hold camera. The target object in the sequence is captured with scale change, rotation, fast motion and partial occlusion.

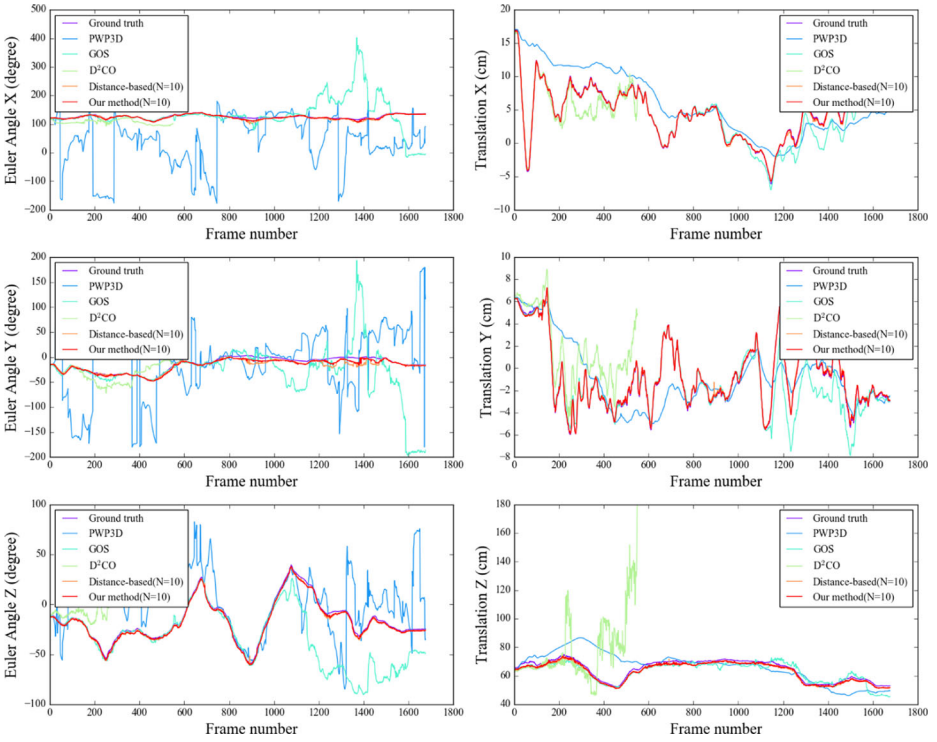


Fig. 11 Comparison of the estimated pose parameters for DUCK between our method($N = 10$) with ground truth, PWP3D, GOS, D^{2}CO and distance-based method. The estimated pose parameters of our method are closer to ground truth than PWP3D, GOS, D^{2}CO and distance-based method

Table 1 gives the time and accuracy performance of our method. The proposed method runs 9fps with 10 particles and 0.9fps with 100 particles on average. For tracking accuracy, it achieves 4.3° rotation error and 1.7cm translation error, and 1.5cm average distance with 100 particles on average. In Fig. 11 we take the DUCK sequence for example to compare the pose parameters(the Euler rotation vector and the translation vector) obtained by our method($N = 10$) with the ground truth, PWP3D, GOS, D^2CO and distance-based method. Our method is superior to the others, and the recovered pose parameters are closer to the ground truth generally. In addition, Fig. 12 gives the tracking results of the 4 sequences by our method($N = 10$).

7 Limitations and future work

Our method directly optimizes the pose parameters in edge distance field, thus it depends on the edge map of the query image. If the color of background is similar to the object, or severe motion blur occurs in the scene, we cannot get adequate contours of the target. Our method will fail because it merely exploits the contour information. We assume that the occluded or mismatched 3D contour points tend to have a lower direction consistency score. Actually it is not enough to track the pose only by contour especially when similar edges appear in the scene, the information of the inner pixels should also be exploited to alleviate the ambiguities from background. In future work, we will consider the information of the inner pixels to enhance the tracking robustness.

For full symmetrical objects (such as spheres and cylinders), multiple poses may result into the same contour, which is ambiguous for our method to estimate the pose correctly.

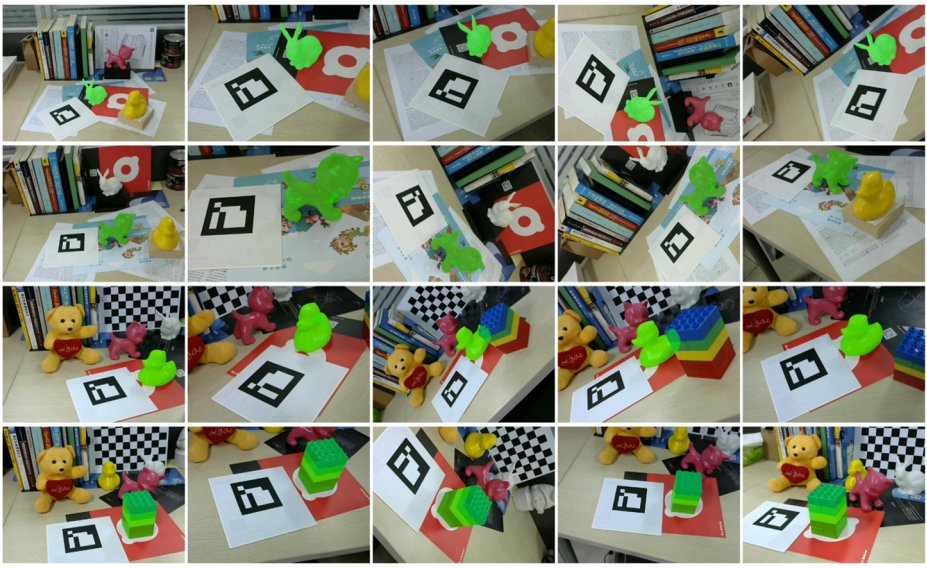


Fig. 12 Tracking results by our method with 10 particles for BUNNY, CAT, DUCK and LEGO. The squared marker in the scene is only used to provide the ground truth pose of the target object for quantitative evaluation

Moreover, the computation cost is also a critical problem for fast object tracking using massive particles. We will speed up the particle filtering using GPU technique.

8 Conclusions

In this paper, we propose an edge-based 3D tracking approach without explicit 3D-2D correspondences which performs robustly in real environments. Our method exploits both distance and direction, with distance for pose estimation and direction for pose validation. The distance-based pose estimation is fast and accurate with a good initialization, the estimated pose then is validated with edge direction for better robustness. We incorporate direction-based validation with robust pose estimation, non-local searching and failure recovery. Comparative experiments demonstrate that the proposed method is effective on real image sequences with occlusions, large motions and background clutters.

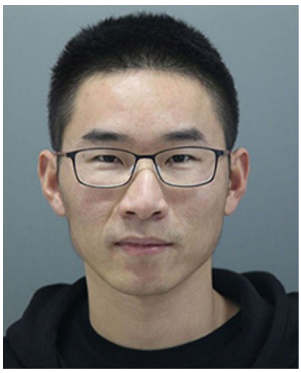
Acknowledgements The authors gratefully acknowledge the anonymous reviewers for their comments to help us to improve our paper, and also thank for their enormous help in revising this paper. This work is supported by the National Key Research and Development Program of China (No. 2016YFB1001501).

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. Besl PJ, Mckay ND (1992) Method for registration of 3-D shapes. In: Robotics - DL tentative, pp 239–256
2. Calonder M, Lepetit V, Strecha C, Fua P (2010) BRIEF: Binary robust independent elementary features. In: European conference on computer vision, pp 778–792
3. Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 8(6):679–98
4. Chen Y, Medioni G (1992) Object modelling by registration of multiple range images. Image Vis Comput 10(3):145–155
5. Choi C, Christensen HI (2012) 3D textureless object detection and tracking: an edge-based approach. In: International conference on intelligent robotics and system, pp 3877–3884
6. Choi C, Christensen HI (2012) Robust 3D visual tracking using particle filtering on the special Euclidean group: A combined approach of keypoint and edge features. Int J Robot Res 33(4):498–519
7. Comport AI, Marchand E, Chaumette F (2003) A real-time tracker for markerless augmented reality. In: IEEE International symposium on mixed and augmented reality, pp 36–45
8. Felzenszwalb PF, Huttenlocher DP (2004) Distance transforms of sampled functions. Theory of Computing 8(19):415–428
9. Fitzgibbon AW (2003) Robust registration of 2D and 3D point sets. Image Vis Comput 21(13):1145–1153
10. Gao X, Hou XR, Tang J, Cheng HF (2003) Complete solution classification for the perspective-three-point problem. IEEE Trans Pattern Anal Mach Intell 25(8):930–943
11. Harris C, Stennett C (1990) RAPId - a video-rate object tracker. In: British machine vision conference, pp 73–77
12. Hartley R, Zisserman A (2000) Multiple view geometry in computer vision. Cambridge University Press
13. Hinterstoisser S, Cagniart C, Ilic S, Sturm P, Navab N, Fua P, Lepetit V (2012) Gradient response maps for real-time detection of textureless objects. IEEE Trans Pattern Anal Mach Intell 34(5):876–888
14. Hinterstoisser S, Lepetit V, Ilic S, Fua P, Navab N (2010) Dominant orientation templates for real-time detection of texture-less objects. In: IEEE Conference on computer vision and pattern recognition, pp 2257–2264

15. Hinterstoisser S, Lepetit V, Ilic S, Holzer S, Bradski G, Konolige K, Navab N (2012) Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In: Asian conference on computer vision, pp 548–562
16. Imperoli M, Pretto A (2015) D^2CO: Fast and robust registration of 3D textureless objects using the directional chamfer distance. In: International conference on computer vision systems, pp 316–328
17. Isard M, Blake A (1998) CONDENSATION - Conditional density propagation for visual tracking. In: International journal of computer vision, pp 5–28
18. Kato H, Billinghurst M (1999) Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: IEEE And ACM international workshop on augmented reality, pp 85–94
19. Klein G, Murray DW (2006) Full-3D edge tracking with a particle filter. In: British machine vision conference, pp 1119–1128
20. Lepetit V, Fua P (2005) Monocular model-based 3D tracking of rigid objects: a survey. Found Trends Comput Graph Vis 1(1):1–89
21. Lepetit V, Moreno-Noguer F (2009) Epnp: an accurate o(n) solution to the pnp problem. Int J Comput Vis 81(2):155–166
22. Liu MY, Tuzel O, Veeraraghavan A, Chellappa R (2010) Fast directional chamfer matching. In: IEEE Conference on computer vision and pattern recognition, pp 1696–1703
23. Lourakis M, Zabulis X (2013) Model-based pose estimation for rigid objects. In: International conference on computer vision systems, pp 83–92
24. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110
25. Marchand E, Bouthemy P, Chaumette F (2001) A 2Dc3D model-based approach to real-time visual tracking. Image Vis Comput 19(13):941–955
26. Marchand E, Uchiyama H, Spindler F (2015) Pose estimation for augmented reality: a hands-on survey. IEEE Trans Vis Comput Graph 22(12):2633–2651
27. Park Y, Lepetit V, Woo W (2008) Multiple 3D object tracking for augmented reality. In: IEEE International symposium on mixed and augmented reality, pp 117–120
28. Park Y, Lepetit V, Woo W (2011) Texture-less object tracking with online training using an rgb-d camera. In: IEEE International symposium on mixed and augmented reality, pp 121–126
29. Prisacariu VA, Reid ID (2012) PWP3D: Real-time segmentation and tracking of 3d objects. Int J Comput Vis 98(3):335–354
30. Rublee E, Rabaud V, Konolige K, Bradski G (2011) ORB: An efficient alternative to SIFT or SURF. In: IEEE International conference on computer vision, pp 2564–2571
31. Seo BK, Park H, Park JI, Hinterstoisser S, Ilic S (2014) Optimal local searching for fast and robust textureless 3D object tracking in highly cluttered backgrounds. IEEE Trans Vis Comput Graph 20(1):99–110
32. Shotton J, Glocker B, Zach C, Izadi S, Criminisi A, Fitzgibbon A (2013) Scene coordinate regression forests for camera relocalization in rgb-d images. In: IEEE Conference on computer vision and pattern recognition, pp 2930–2937
33. Vacchetti L, Lepetit V, Fua P (2004) Combining edge and texture information for real-time accurate 3D camera tracking. In: IEEE International symposium on mixed and augmented reality, pp 48–56
34. Vacchetti L, Lepetit V, Fua P (2004) Stable real-time 3D tracking using online and offline information. IEEE Trans Pattern Anal Mach Intell 26(10):1385–1391
35. Wang B, Zhong F, Qin X (2017) Pose optimization in edge distance field for textureless 3d object tracking. In: The computer graphics international conference, pp 1–6
36. Wang G, Wang B, Zhong F, Qin X, Chen B (2015) Global optimal searching for textureless 3D object tracking. Vis Comput 31(6):979–988
37. Wuest H, Vial F, Strieker D (2005) Adaptive line tracking with multiple hypotheses for augmented reality. In: IEEE International symposium on mixed and augmented reality, pp 62–69



Bin Wang is now a Ph.D. Candidate of School of Computer Science and Technology in Shandong University. He received his B.Sc. from Shandong University in 2012. His research interests include object tracking and pose estimation, etc.

Fan Zhong is now an associate professor of School of Computer Science and Technology, Shandong University. He received his Ph.D. from Zhejiang University in 2010. His research interests include image and video editing, and augmented reality, etc.

Xueying Qin is a professor of School of Software, Shandong University. She received her Ph.D. from Hiroshima University of Japan in 2001, and M.S. and B.S. from Zhejiang University and Peking University in 1991 and 1988, respectively. Her main research interests are augmented reality, video-based analyzing, and photorealistic rendering, etc.