# Robust 6-DoF Object Pose Tracking with Built-In Recovery under Occlusions and Rapid Object Motions

Balázs Opra Léo Ghafari Tom Stewart Cyrill Stachniss

***Abstract*— Real-time 6-DoF object pose tracking is essential for many robotics applications and a well investigated problem. Yet it remains unreliable under temporary full occlusions and rapid object motions. Once tracking is lost, most methods struggle to detect the failure and recover automatically, often requiring costly reinitialization. In this paper, we address the problem of robust model-based 6-DoF tracking of unseen objects from RGB-D data, especially in scenarios with occlusion and fast motion. We propose a novel method that combines efficient learning-based keypoint matching with optimization-based alignment and introduces a novel failure detection and recovery module. Our system monitors pose reliability, detects divergence or occlusion, and performs a global re-detection and pose estimation step that robustly verifies candidates before resuming tracking. Our evaluation on standard tracking benchmarks and on a new dataset of occluded and fast-moving scenes, which we also release, shows that our method matches state-of-the-art accuracy on easy tracking sequences but runs faster, and provides by far the most robust tracking performance under challenging conditions. Thus, we believe that our approach is a relevant step forward in robust 6-DoF object tracking from RGB-D data.**

## I. Introduction

Model-based object tracking using RGB-D videos has seen significant progress in recent years, with state-of-the-art methods [25], [34] starting to saturate classical benchmarks such as YCB-Video [35]. These methods enable robotics applications to perform manipulation tasks with higher precision, and to be more reactive to dynamic events such as grasping failures or in interactions with humans. Object tracking also serves as the basis for some real-world deployments of robot learning methods [8], [20]. However, full or partial occlusions and sudden object motions, such as those that occur when objects are dropped or temporarily go out of frame, are part of real-world robotics operations and are often not well addressed by object tracking methods.

In this paper, we investigate the problem of accurate and robust model-based object pose tracking under challenging occluded and dynamic scenarios. Model-based object tracking is traditionally addressed by establishing correspondences between an RGB-D video stream and the object model, and framing the tracking problem as iterative pose refinement with numerical optimization [25]. More recently, learning-based methods became the prominent choice for correspondence estimation [22], and partially replaced numerical optimization with a feedforward neural network [34]. Both, optimization and learning-based approaches are capable of highly accurate tracking in slower-moving scenes with partial occlusions, but tend to fail when confronted with fully losing view of the target object or rapid motions. Most methods are unable to effectively recover tracking once diverged and require a full reinitialization.

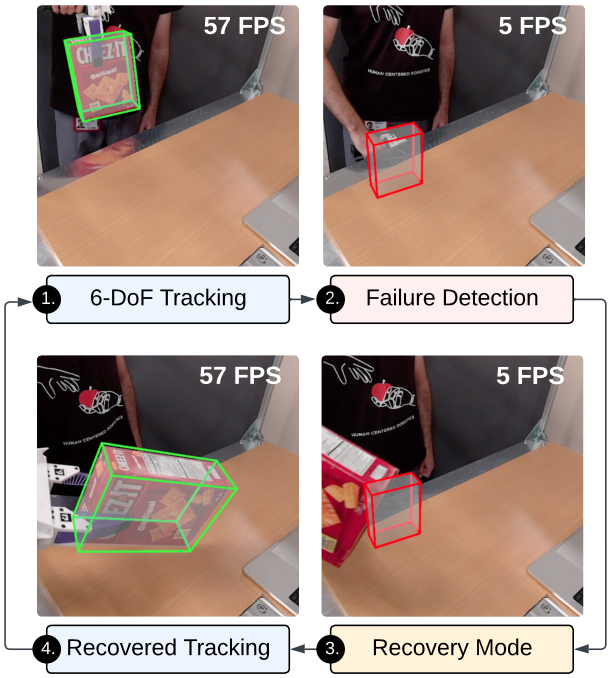


Fig. 1: Our 6-DoF object pose tracking system is capable of detecting tracking failure including due to full occlusions, and is able to re-detect the object and resume tracking automatically.

The main contribution of this paper is a novel accurate model-based RGB-D object pose tracking method, which is uniquely able to detect tracking failure due to divergence or complete occlusion. It is capable of autonomously recovering object tracks once the tracked object becomes visible again. Our system is moreover more efficient than state-of-the-art methods while matching their accuracy on standard benchmarks. We make three key claims: Our approach is able to (i) reliably detect and recover from tracking failure due to divergence or full occlusions; (ii) match the state-of-the-art tracking accuracy on general and robotics-focused datasets; (iii) achieve significantly faster tracking performance than state-of-the-art methods. These claims are backed up by the paper and our experimental evaluation.

Balázs Opra, Léo Ghafari, and Tom Stewart are with Woven by Toyota, Inc. Balázs Opra is also with the University of Bonn, Germany. Cyrill Stachniss is with the University of Bonn, Center for Robotics, and the Lamarr Institute for Machine Learning and Artificial Intelligence. Email: {balazs.opra, leo.ghafari, tom.stewart}@woven.toyota, cyrill.stachniss@igg.uni-bonn.de

## II. Related Work

The classical paradigm for tracking a known 3D object model is to continuously optimize its pose to align with sensor data. Region-based methods like PWP3D [19] segment the object's silhouette and maximize foreground-background separation. Subsequent works improve robustness via localized color models around contours [28], incorporating photometric constraints [37], and efficient sparse updates [26]. Region-based methods excel for texture-poor objects, but can be slow and struggle with axially symmetric objects.

With RGB-D input, many trackers use dense geometric alignment, iterative closest point (ICP) being a common tool for aligning the object's 3D model to depth data [9]. Other methods use probabilistic alignment using signed distance fields [21]. Many works combine modalities such as region and depth [9], [27].

Texture keypoint matching is another common approach, yielding 2D-3D correspondences for pose estimation. Before the widespread adoption of deep learning, methods matched corners [5], [24], or more complex image features such as SIFT [13] to compute frame-to-frame pose updates [10], [25], [29]. Recent works integrate learned components or end-to-end models for tracking. Some employ deep networks to iteratively refine pose estimates [12], or combine learned codes with traditional methods such as particle filters [3]. Neural methods like se(3)-TrackNet [32] use synthetic data to train a neural network to directly predict pose changes between frames. Many learning-based trackers require extensive per-object training which limits their applicability in practice. Other works aim to track objects without an exact CAD model, relying on category priors or online reconstructions [30], [31], [33] at the cost of heavier computation or slight accuracy trade-offs.

The latest trend in object tracking is to train or exploit foundation models for pose estimation. FoundationPose [34] leverages large synthetic datasets to train a model capable of pose estimation and tracking of unseen objects given a CAD model or even a few reference RGB-D images. Other methods use pretrained 3D foundation models such as DINOv2 [16] and MASt3R [11] for pose estimation [2], [17] and tracking [14] using sim2real matching between pre-rendered and real images of the target object.

In our approach, we assume that a CAD model or a reconstructed mesh is available for the target object. Our contribution is a robust optimization-based tracking system with an automatic failure detection and reinitialization module that uses single-frame pose estimation to recover from tracking loss. The system includes intelligent keyframe management and a 3D point-to-point alignment term from depth, both of which improve accuracy in regular tracking scenarios. It yields a robust tracker that outperforms state-of-the-art approaches on challenging tracking scenarios.

## III. Our Approach to Model-Based Pose Tracking

Our system works with one or more RGB-D sensors, and assumes calibrated extrinsic parameters, that is, the homogeneous transformation ${}_{\mathrm{W}}\boldsymbol{T}_{\mathrm{C}}$ between the world frame $\mathrm{W}$ and the camera frame $\mathrm{C}$ is known. We also assume calibrated camera intrinsics. Given an initial pose estimate of the target object model frame $\mathrm{M}$, ${}_{\mathrm{W}}\boldsymbol{T}_{\mathrm{M},0} = {}_{\mathrm{W}}\boldsymbol{T}_{\mathrm{C}}\,{}_{\mathrm{C}}\boldsymbol{T}_{\mathrm{M},0}$, our task is to iteratively update the pose with each incoming RGB-D frame. To estimate the updated pose for a new frame, we use correspondence data from region, texture, and depth modalities.

Our approach builds upon ICG+ [25], a multi-modal RGB-D tracker that combines region, depth, and texture correspondences in a single Gauss-Newton optimization. The region and depth modalities are used *as is*. We however, substantially extend the texture modality of ICG+ with a new learning-based feature extraction and matching pipeline, expanded keyframe management, a 3D residual for optimization, and an integrated failure detection and recovery module. These additions turn the texture subsystem into the core driver of robustness in our tracker. We introduce the overall tracking system and detail the texture-based tracking subsystem. Then, we discuss the failure detection and recovery mechanism in detail.

### A. Optimization-based Tracking

The tracker system is composed of three modalities: region, depth, and texture. They are used to establish correspondences $\mathcal{D}_{\mathrm{r}}, \mathcal{D}_{\mathrm{d}}, \mathcal{D}_{\mathrm{t}}$ between the tracked object's body of the previous tracking iteration at $t-1$ and the current color and depth data coming from one or more RGB-D cameras. The exact form of $\mathcal{D}$ depends on the specific modality. The depth modality uses model point to depth point correspondences, the region modality derives correspondences by considering image pixels along correspondence lines which are orthogonal to the model body, and the texture modality derives correspondences from image keypoint matching.

In each tracking iteration, we aim to use the correspondences to find a small pose variation $\boldsymbol{\theta}^{\mathsf{T}} = \begin{bmatrix}\boldsymbol{\theta}_r^{\mathsf{T}} & \boldsymbol{\theta}_t^{\mathsf{T}}\end{bmatrix}$ which is composed of the rotation vector $\boldsymbol{\theta}_r \in \mathbb{R}^3$ and translation vector $\boldsymbol{\theta}_t \in \mathbb{R}^3$, to update the estimated object pose. Specifically, once we found $\boldsymbol{\theta}$ using the latest correspondences, we update the pose estimate as

$$ {}_{\mathrm{W}}\boldsymbol{T}_{\mathrm{M},\,t+1} = {}_{\mathrm{W}}\boldsymbol{T}_{\mathrm{M},\,t} \begin{bmatrix} \boldsymbol{I} + [\boldsymbol{\theta}_r]_\times & \boldsymbol{\theta}_t \\ \boldsymbol{0} & 1 \end{bmatrix}, \tag{1} $$

based on the first-order approximation of the exponential map on $\mathrm{SE}(3)$.

We find $\boldsymbol{\theta}$ by maximizing the posterior probability density function

$$ p(\boldsymbol{\theta} \mid \mathcal{D}) = \prod_{i=1}^{n_{\mathrm{r}}} p(\boldsymbol{\theta} \mid \mathcal{D}_{\mathrm{r}i}) \prod_{i=1}^{n_{\mathrm{d}}} p(\boldsymbol{\theta} \mid \mathcal{D}_{\mathrm{d}i}) \prod_{i=1}^{n_{\mathrm{t}}} p(\boldsymbol{\theta} \mid \mathcal{D}_{\mathrm{t}i}), \tag{2} $$

where $n_{\mathrm{r}}, n_{\mathrm{d}}, n_{\mathrm{t}}$ are the number of region, depth, and texture correspondences, respectively. When multiple RGB-D sensors are available, separate modality instances can be configured per camera and all contribute correspondences to the joint optimization, following ICG+.

To find a $\hat{\boldsymbol{\theta}}$ that maximizes Eq. (2), we use Gauss-Newton optimization with Tikhonov regularization,

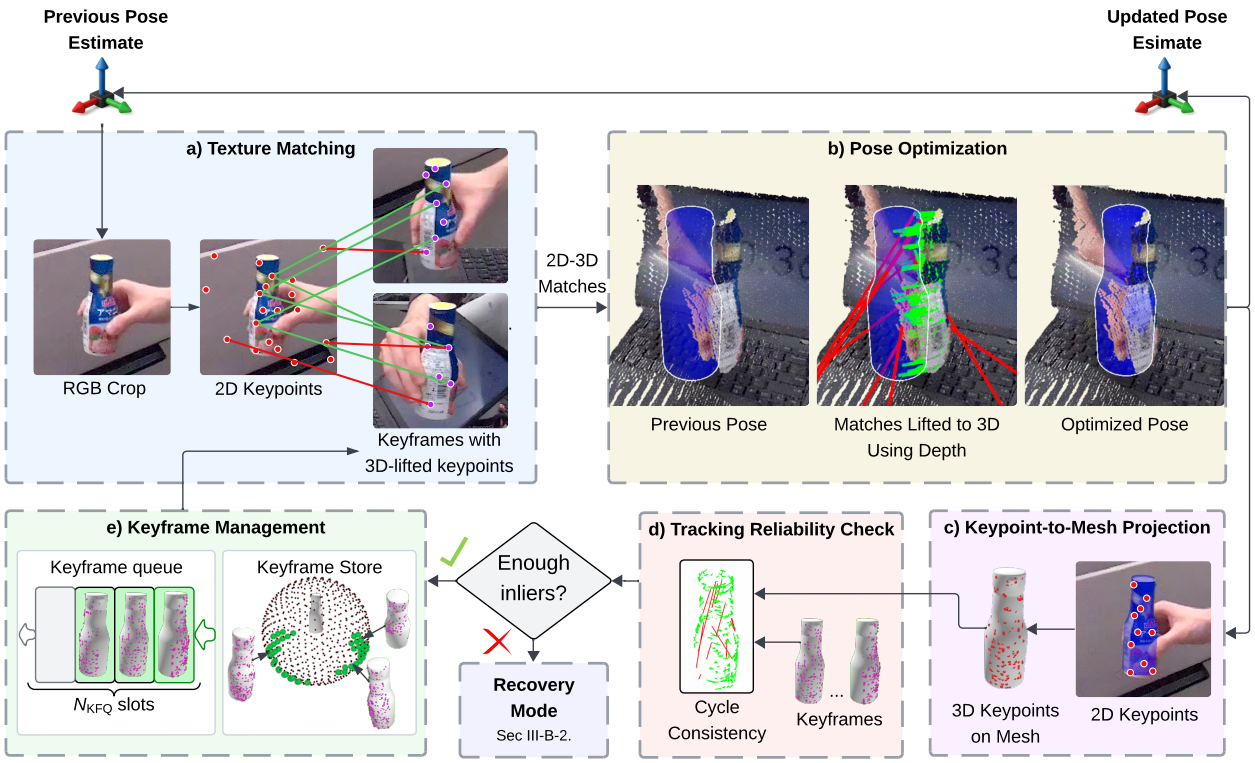


Fig. 2: Overview of the Texture-Based Tracking System. (a) We perform texture-based keypoint matching between frozen prior frames called keyframes and the current frame. (b) We lift the matches to 3D using the object model and depth data, filter outliers, and update the pose estimate via numerical optimization. (c) Using the updated pose, we project the image keypoints onto the model, and (d) perform a cycle-consistency-based reliability check. (e) If the check passes, we store the current frame as a keyframe for future matching.

$$\hat{\boldsymbol{\theta}} = (-\boldsymbol{H} + \boldsymbol{\Lambda})^{-1}\ \boldsymbol{g},\ \boldsymbol{\Lambda} = \begin{bmatrix} \lambda_R\, \boldsymbol{I}_3 & \boldsymbol{0} \\ \boldsymbol{0} & \lambda_t\, \boldsymbol{I}_3 \end{bmatrix} \tag{3}$$

where the gradient $\boldsymbol{g}$ and the Hessian $\boldsymbol{H}$ are the first- and second-order derivatives of the logarithm of the PDF, and $\lambda_R$ and $\lambda_t$ are the rotation and translation Tikhonov regularization parameters. Intuitively, the Tikhonov regularization helps stabilize the optimization in cases where any of the degrees of freedom of the pose are underconstrained.

We use *unmodified* ICG+ [25] implementations for region and depth modalities and refer the reader to that paper for details. In the next section, we describe our novel texture-based tracking modality approach.

### B. Texture Modality

In the texture modality, we use local image feature matching to establish correspondences. At the beginning of each tracking iteration, we project the tracked object's model into the color image of each RGB-D sensor using the latest pose estimate. We then crop a square-shaped region of interest around the projected object model, and extract image features from it. To take advantage of advances in efficient learning-based image matching, we use the SuperPoint [4] feature detector network with an efficient NVIDIA TensorRT [38] implementation. This allows us to use input images of size $1024\times1024$ and set the number of keypoints to 1800.

Subsequently, we match the features with those of so-called keyframes saved from previous frames. A keyframe is a frame that is kept static, and we match subsequent frames against it. We create new keyframes as the object moves a certain amount since the last keyframe, and describe our keyframe management in detail in the next section. We use nearest-neighbor matching with $L2$ distance as our matching algorithm, and keep matches that pass both, Lowe's ratio test [13] and the two-view cycle-consistency check [6]. We implement the nearest-neighbor search with LibTorch [18] on the GPU, which allows us to maintain efficiency despite the increased number of keypoints. Each match produces a three- or two-dimensional residual through either 3D point distance or reprojection error, and contributes to the gradient and Hessian; further details are given in Sec. III-B.3.

*1) Keyframe management:* Following Stoiber et al. [25], we create a new keyframe during tracking if the orientation of the object ${}_{\mathrm{W}}\boldsymbol{R}_{\mathrm{M}} \in \mathbf{SO}(3)$ has changed by more than $10°$ since the last keyframe. We maintain a queue of keyframes of size $N_{\mathrm{KFQ}}$, dropping the oldest keyframe if the queue is full.

**Keyframe creation.** To create a keyframe, we use the latest object pose estimate in the camera frame, ${}_{\mathrm{C}}\boldsymbol{T}_{\mathrm{M}} = {}_{\mathrm{W}}\boldsymbol{T}_{\mathrm{C}}^{-1}\,{}_{\mathrm{W}}\boldsymbol{T}_{\mathrm{M}}$, to project the detected texture keypoints onto the surface of the model. Specifically, we transform the model into the camera frame using ${}_{\mathrm{C}}\boldsymbol{T}_{\mathrm{M}}$, and for each keypoint location we find the intersection of the camera ray with the transformed model using the known intrinsic parameters. The keyframe stores these intersection points defined in the model frame, ${}_{\mathrm{M}}\boldsymbol{X}$, together with their visual

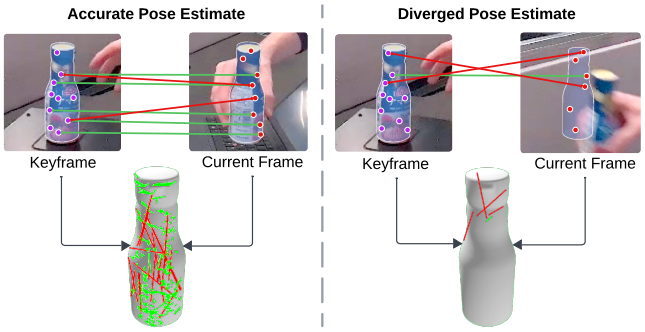


Fig. 3: Tracking reliability check. The keyframe-current frame matches are projected into the model frame using the current pose estimate. Left: inlier (green) and (outlier) matches when the current pose is accurate. Right: very few matches and inliers when the pose estimate diverged.

feature descriptor vectors.

Note that not all detected keypoints from the region of interest fall onto the surface of the model, as some may correspond to the background or lie on an occluded part of the object. As in ICG+ [25], we use the depth image to filter out occluded keypoints, and only keep those that fall on the unoccluded surface of the object model. In each tracking iteration we match all keyframes in the keyframe queue with the current frame's region of interest.

**Keyframe store.** However, only using a queue of the most recent keyframes is insufficient for robust tracking under fast object motions and for recovery from tracking failure, as previously observed informative views of the object go out of memory as the object rotates. Thus, we also maintain a buffer called keyframe store where we permanently store keyframes from already observed view directions. The view direction slots of this buffer are represented by the vertices of a sub-sampled icosahedron surrounding the target object, as shown in Fig. 2. At the end of each tracking iteration, we replace the keyframe in the slot with the closest view direction to the current object pose if (a) the slot is empty, or (b) if the current frame has more keypoints than the stored keyframe. The keyframe store allows the system to maintain a persistent map of the object's appearance from all previously observed viewpoints, regardless of how much the object has moved since. This is particularly useful for fast-moving objects, as well as for redetecting an object after tracking failure.

Thus, in addition to the frames in the keyframe queue, we also select up to $N_{\text{KFS}}$ keyframes from the keyframe store for matching during tracking. These are selected from a 30° view cone around the current view direction of the object, while maintaining a minimum distance of 5° from all other selected keyframes.

*2) Tracking reliability check:* Since keyframes are created by projecting 2D image keypoints onto the model body using the current pose estimate, an inaccurate estimate will lead to projected keypoints that do not correspond to their real-world locations on the model. To avoid creating inaccurate keyframes, we perform a reliability check using the matches established between the current frame's region of interest and all active keyframes.

Specifically, at the end of a tracking iteration we find all matched, unoccluded keypoints in the current frame which, when projected onto the body mesh using the updated pose estimate, lie on the surface of the body mesh. We then perform a cycle-consistency check for all such keypoints, noting that if (a) the image feature matches, and (b) the prior keyframe's pose estimate, and (c) the current pose estimate are correct, then the matched keypoints should lie at the same 3D position in the model coordinate system in both, the keyframe and the current frame. Concretely, we deem keypoint matches inliers if we have the following:

$$\left\| {}_{\text{M}}\boldsymbol{X}^{\text{keyframe}} - {}_{\text{M}}\boldsymbol{X}^{\text{current}} \right\|_2 < \epsilon_{\text{cc}},\ \epsilon_{\text{cc}} \in \mathbb{R}^+ \tag{4}$$

If there is at least one keyframe with at least $N_{\text{inliers}}$ inlier keypoints, we consider the current pose estimate acceptable and allow a keyframe to be created from the current frame. An appropriate value for $N_{\text{inliers}}$ depends on the number of keypoints we detect. We found $N_{\text{inliers}} = 10$ to work well in all our experiments. If there is more than one configured camera, we perform this check for all cameras individually and allow or disallow the creation of keyframes per camera. If none of the cameras pass the reliability check, we consider the tracking to have failed and explicitly begin the tracking recovery procedure, described in Sec. III-C.

*3) Pose probability density function:* We use two PDFs of pose variation based on different residuals: a 3D–3D distance residual when depth is available and a 2D reprojection residual otherwise. If the depth image contains a valid depth measurement at a matched keypoint location, we reconstruct the 3D point ${}_{\text{D}}\boldsymbol{P}$ from it in the depth sensor frame and move it into the body frame using the current pose estimate and the pose variation vector:

$${}_{\text{M}}\boldsymbol{P}(\boldsymbol{\theta}) = {}_{\text{M}}\boldsymbol{T}(-\boldsymbol{\theta}) {}_{\text{M}}\boldsymbol{T}_{\text{D}} {}_{\text{D}}\boldsymbol{P} \tag{5}$$

$$\text{with } {}_{\text{M}}\boldsymbol{T}(-\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{I} + [-\boldsymbol{\theta}_r]_\times & -\boldsymbol{\theta}_t \\ \boldsymbol{0} & 1 \end{bmatrix}. \tag{6}$$

We use the 3D distance between ${}_{\text{M}}\boldsymbol{P}(\boldsymbol{\theta})$ and the matching 3D model point of the keyframe, ${}_{\text{M}}\boldsymbol{X}$, as the residual. We assume a normal distribution for the residuals and formulate the PDF for $n$ independent point pairs as

$$p(\boldsymbol{\theta}|\mathcal{D}_{\text{ti}}) \propto \prod_{i=1}^{n} \exp\left( -\frac{1}{2\sigma^2 \text{d}^2} \rho_{\text{tuk}}\left(r_i\right) \right) \tag{7}$$

$$\text{with } r_i = \left\| {}_{\text{M}}\boldsymbol{X}_i - {}_{\text{M}}\boldsymbol{P}_i(\boldsymbol{\theta}) \right\|_2. \tag{8}$$

We scale the residuals by the inverse of the square of $\text{d}$, the depth of the correspondence point, which is proportional to the uncertainty of the depth measurement. Furthermore, the user-defined variance $\sigma^2$ is used to balance the influence of the texture modality with respect to the other modalities.

To reduce the impact of outliers, we use Tukey's biweight loss

$$\rho_{\text{tuk}}(x) = \begin{cases} \frac{c^2}{6}(1 - (1 - (\frac{x}{c})^2)^3) & |x| \le c \\ \frac{c^2}{6} & |x| > c, \end{cases} \tag{9}$$

where $c$ is a user-defined threshold, ideally selected based on the expected noise level of the measurements.

To derive the gradient and Hessian, we first reformulate the Tukey loss as an iteratively reweighted least squares problem

$$\rho_{\text{tuk}}\left(r_i\right) \approx w_i r_i^2, \tag{10}$$

where the weight is calculated as $w_i = \rho_{\text{tuk}}\left(r_i\right)/r_i^2$, evaluating the residuals at the current pose estimate with $\boldsymbol{\theta} = \mathbf{0}$. With this, we can compute the gradient and Hessian using the Gauss-Newton approximation as

$$\boldsymbol{g}_{\text{t}} = -\frac{w_i}{\sigma^2 \text{d}^2}\left({}_{\text{M}}\boldsymbol{X} - {}_{\text{M}}\boldsymbol{P}\right)^{\text{T}}\left[-\left[{}_{\text{M}}\boldsymbol{P}\right]_{\times} \quad \boldsymbol{I}_3\right] \tag{11}$$

$$\boldsymbol{H}_{\text{t}} \approx -\frac{w_i}{\sigma^2 \text{d}^2}\begin{bmatrix}\left[{}_{\text{M}}\boldsymbol{P}\right]_{\times} \\ \boldsymbol{I}_3\end{bmatrix}\left[\left[-{}_{\text{M}}\boldsymbol{P}\right]_{\times} \quad \boldsymbol{I}_3\right]. \tag{12}$$

Lifting the 2D-3D correspondences to 3D-3D by using the depth information enables us to filter outlier matches more effectively. Matches between the target object and a nearby background pixel may have a small reprojection error in the image space, but a large 3D distance error. Although the Tukey loss helps reduce the influence of such outliers, we find that discarding all matches with a 3D distance error larger than a threshold $\epsilon_{3\text{D}} > c$ improves accuracy even further. A constant value of $\epsilon_{3\text{D}} = 2c$ works well in all our experiments.

When there is no valid depth measurement at the keypoint location, we fall back to using the reprojection error as described by Stoiber et al. [25]. Refer to Fig. 2 for an overview of the texture-based tracking system.

### C. Failure Recovery

Our recovery module is integrated into the control flow of the tracker: the reliability check (Sec. III-B) detects divergence or occlusion and triggers a global re-detection step. Recovery performs exhaustive crop-to-keyframe matching, lifts matches to 3D using depth, solves robust point cloud registration to obtain candidate object poses, and resumes tracking only after verifying the candidate pose using the reliability check. This design avoids any external error detection. In case the tracking fails, recovery is needed.

We do not assume any pose prior after a tracking failure, as the object may have moved rapidly or may have been temporarily occluded. Thus, instead of focusing only on the region of interest around the last pose estimate, we divide the entire color image into minimally overlapping crops sized $0.25$ times the longer side of the image, to be used for matching. We have found that this crop size works well for most camera configurations used in robotic manipulation, and produces $N_{\text{crops}} = 16$ crops for a $1280\times1024$-pixel image. We similarly do not use any pose prior for selecting keyframes, and instead use a strategy aimed at maximizing the view direction coverage of the object. In particular, we use farthest-point sampling to select $N_{\text{recovery}}$ keyframes from the keyframe store whose view direction vectors are maximally distant from each other.

We then extract SuperPoint features from the crops and match each crop with all of the selected keyframes using the same matching strategy as described in Sec. III-B. This gives us a set of 2D-3D crop-to-model point correspondences for each crop. We then use the depth information to reconstruct the 3D points from the crops to create 3D-3D correspondences between the depth point cloud and the object model. We solve the resulting point cloud registration problems using TEASER++ [36], which is a fast and robust registration method, to obtain $N_{\text{crops}} \times N_{\text{recovery}}$ pose estimates. The best candidate pose is selected as the one with the highest number of inlier keypoints retrieved from the registration solution.

Since the recovery pose proposals are coarse, we reinitialize tracking from the best proposal pose estimate and run a full tracking iteration, which acts as a pose refinement step. We then perform the tracking reliability check and exit recovery mode and continue tracking from the recovered pose estimate if it passes. If the reliability check fails, we discard the result of the recovery attempt and continue repeating the recovery procedure for each new frame until the tracking reliability check passes. The reliability check helps avoid spuriously accepting incorrect recovery pose proposals.

During recovery, instead of only matching a single image crop around the last pose estimate, we have to match multiple crops of the current frame with keyframes, and have to additionally solve a point cloud registration problem for each of these crops. This makes computing the recovery pose estimate significantly more expensive than a regular tracking iteration. In our experiments we had $N_{\text{crops}} = 16$ and set $N_{\text{recovery}} = 20$, and we found that it takes about $210\,\text{ms}$ on average to compute the recovery pose estimate, which is around 12 times more than a regular tracking iteration. Despite this, the recovery procedure is still significantly faster than a nominal full object re-detection and pose estimation pipeline. For example, a pipeline composed of the known object segmentation method CNOS [15] and the pose estimation method FoundationPose [34] takes several seconds to run, and requires additional engineering effort to implement.

At this point, we want to formulate the difference to ICG+ explicitly. Our method introduces several major extensions: (i) high-resolution SuperPoint+ TensorRT feature extraction and GPU matching, enabling an order of magnitude more keypoints in real time; (ii) a 3D distance residual with analytic gradient/Hessian, improving optimization accuracy; (iii) expanded keyframe management, including an icosahedron-indexed keyframe store that preserves viewpoints beyond the recency-based queue; (iv) a cycle-consistency-based reliability check that detects divergence or occlusion; and (v) an integrated recovery procedure that globally re-detects the object and robustly verifies recovery candidate poses. Our changes improve tracking accuracy especially for axially symmetric objects and greatly increase robustness in fast-moving and occluded scenarios, as shown by our evaluation in Sec. IV.

## IV. Experimental Evaluation

The main focus of this work is an RGB-D object pose tracking method that is capable of accurately and robustly

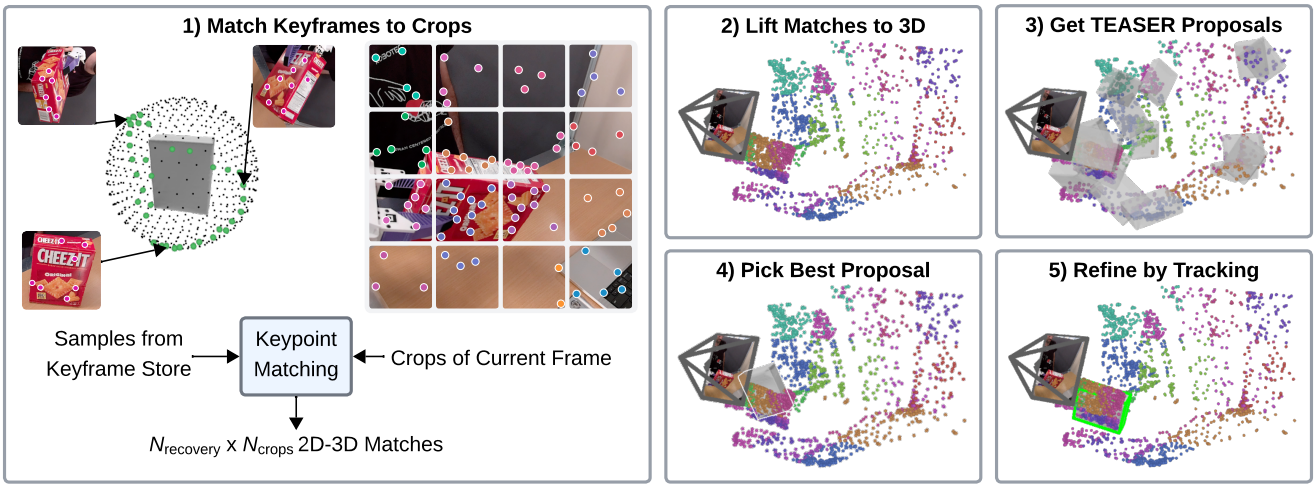


Fig. 4: Failure recovery procedure: 1) we match well-selected keyframe store keyframes with all crops of the current image, 2) these matches are lifted to 3D using depth data, and 3) the TEASER point cloud registration method computes pose proposals for each pair. 4) The proposal with the most TEASER inliers is 5) refined by one iteration of tracking and verified by the reliability check.

tracking objects with known 3D models.

We present our experiments to show the capabilities of our method. Our evaluation also supports our key claims, which are: (i) our method can reliably detect and recover from tracking failure due to divergence or full occlusions; (ii) it matches state-of-the-art tracking accuracy on general and robotics-focused datasets; (iii) it achieves significantly faster tracking performance than state-of-the-art methods.

### A. *Experimental Setup*

We evaluated our approach on the YCB-Video [35] and YCBInEOAT [32] datasets, which contain RGB-D videos of YCB objects in various scenarios, including cluttered scenes and robotic manipulation tasks. We also evaluated our approach on a novel RGB-D tracking dataset we collected in our lab and which features rapid object motions and full occlusions. Our dataset contains RGB-D videos of household objects recorded using two RGB-D cameras with overlapping fields of view. Note that only one camera is used for tracking; the other camera is for multi-view verification of our annotations. The objects are manipulated by hand or with the UMI hand-held gripper [1], and include the YCB-V cracker box and mug, and an additional axially-symmetric textured bottle, shown in Fig. 2. We divide the dataset into ”fast” and ”occluded” sequences, where ”fast” sequences contain fast object motions, including those which cause significant motion blur, and ”occluded” sequences where the target object is temporarily fully occluded as it leaves the field of view of the camera. We obtain pseudo-ground truth poses using a state-of-the-art single-view pose estimation method [34] and manually refine them to ensure accuracy and multi-view consistency. We use a learning-based stereo method [23] as our source for depth data.

We compare our approach with three baseline methods. We compare to PoseRBPF [3], a particle filter based 6-DoF tracker that decouples pose estimation into translation and discretized rotation parts, using an autoencoder network to construct a codebook for classifying rotations. Even though this method requires per-object retraining, as opposed to all other evaluated methods which do not require training for new objects, we select it for comparison as we expect that it is more robust to fast motions and tracking loss due to the particle filter formulation. We also compare against ICG+ [25], a multi-camera RGB-D object tracking method of which our method is an extension, and with the state-of-the-art single-view pose estimation and tracking method FoundationPose [34]. To evaluate tracking accuracy, we use the ADD and ADD-S area under curve metrics, which are based on the average distance and average shortest distance error between the model points transformed by the estimated pose and ground truth pose, as proposed by Hinterstoisser et al. [7]. A perfect ADD score requires a fully accurate pose estimate, while ADD-S is more lenient toward rotation errors and is suitable for textureless objects with rotational symmetries.

We also evaluate runtime performance of the methods using a desktop computer with an Intel(R) Core(TM) i9-14900K CPU and an NVIDIA GeForce RTX 4090 GPU.

### B. *Tracking Failure Recovery*

The first experiment evaluates the performance of our approach and its outcomes support the claim that the system is more robust during rapid object motion and occlusions, and is capable of recovering from tracking failure caused by divergence due to fast object motion or full occlusions. For this evaluation, we compute the tracking accuracy metrics on the ”fast” and ”occluded” sequences of our dataset for all four methods. To ensure a fair comparison when performing this evaluation, we emulate the increased runtime of our method in the failure recovery mode. In particular, we delay the recovery pose estimate by 6 frames, freezing the output pose in the meantime, which is the average number of frames it takes to compute the recovery pose estimate in our online measurements, assuming a camera running at 30 FPS.

We present the results in Tab. I showing the per-object, per-sequence category, as well as the overall scores. The

TABLE I: Tracking accuracy of PoseRBPF (PRBPF) [3], ICG+ [25], FoundationPose (FP) [34], and our method our dataset. Results are categorized per object and per setting. The best and second-best results are marked in bold and underlined, respectively.

| Approach | PRBPF | | ICG+ | | FP | | Ours | |
|---|---|---|---|---|---|---|---|---|
| Novel object | ✗ | | ✓ | | ✓ | | ✓ | |
| Metric | ADD | ADD-S | ADD | ADD-S | ADD | ADD-S | ADD | ADD-S |
| bottle | 43.69 | 47.44 | 48.58 | 56.84 | 44.99 | 46.84 | **90.46** | **91.09** |
| mug | 60.57 | 69.04 | 36.29 | 55.32 | 48.19 | 48.65 | **66.09** | **77.33** |
| cracker_box | 42.90 | 68.57 | 45.59 | 53.06 | 43.94 | 43.94 | **92.66** | **93.25** |
| Fast | 65.46 | 74.88 | 46.71 | 64.26 | 44.89 | 46.24 | **88.95** | **92.05** |
| Occluded | 32.95 | 51.33 | 40.50 | 46.00 | 46.23 | 46.25 | **78.70** | **83.47** |
| **All Frames** | 48.69 | 62.74 | 43.51 | 54.84 | 45.58 | 46.25 | **83.67** | **87.63** |

baseline methods achieve lower scores because they lose track of the object during fast motions and only have a limited capability to recover. Our approach also loses track of the object for a number of frames but is able to detect and recover from the failure, achieving significantly stronger tracking accuracy. It achieves the lowest scores on the largely textureless and rotationally symmetric YCB-V mug object, which is intuitively expected for a texture-focused method, yet it still outperforms baselines, highlighting the system's overall robustness. Notably, the older method PoseRBPF achieves the second-best overall accuracy, which we attribute to the particle filter formulation being more robust to fast motions and tracking loss.

### C. Accurate and Fast Tracking

The second experiment evaluates tracking accuracy and runtime performance on the YCB-Video and YCBInEOAT datasets. The results shown in Tab. II support our claim that our approach achieves a tracking accuracy similar to the state-of-the-art method FoundationPose while being around 40% faster on the same hardware. Although the baseline method ICG+ [25] runs faster than our method, it achieves lower tracking accuracy. Our method brings a significant improvement compared to ICG+ especially for rotation accuracy, as shown by the increase in ADD score on the YCBInEOAT sequences featuring the axially symmetric tomato soup can object. YCBInEOAT scenes feature robotic manipulation, where objects often undergo significant rotations in the camera frame, as opposed to the more static YCB-Video scenes.

### D. Ablation Study

The third experiment validates the effectiveness of our proposed 3D texture matching PDF, the use of the SuperPoint keypoint detector, and the keyframe store to improve tracking accuracy. We run this experiment on the YCBInEOAT dataset as it is representative of robotic manipulation scenarios. The different settings and results are shown in Tab. III, where the baseline setting uses 300 SIFT keypoints and one keyframe ($N_{\text{KFQ}} = 1$). Subsequent settings add the 3D residual, switch the keypoint extractor to SuperPoint with 1800 keypoints, increase the number of keyframes to $N_{\text{KFQ}} = 9$, and finally enable the keyframe store with $N_{\text{KFS}} = 10$. The results show that our proposed extensions increase the accuracy in general,

TABLE II: Tracking accuracy and speed on the YCB-Video and YCBInEOAT datasets. The best and second-best all-frame results are marked in bold and underlined, respectively. For per-object results, the best results are marked by a gray highlight.

| Approach | PRBPF | | ICG+ | | FP | | Ours | |
|---|---|---|---|---|---|---|---|---|
| Novel object | ✗ | | ✓ | | ✓ | | ✓ | |
| Metric | ADD | ADD-S | ADD | ADD-S | ADD | ADD-S | ADD | ADD-S |
| **YCB-Video** | | | | | | | | |
| master_chef_can | 90.5 | 95.1 | 94.7 | 98.0 | 93.6 | 97.0 | 96.1 | 98.4 |
| cracker_box | 88.2 | 93.0 | 96.8 | 98.4 | 96.9 | 97.8 | 96.2 | 98.2 |
| sugar_box | 92.9 | 95.5 | 96.6 | 98.5 | 96.9 | 98.2 | 97.5 | 98.9 |
| tomato_soup_can | 90.0 | 93.8 | 95.2 | 98.1 | 96.3 | 98.1 | 95.3 | 98.2 |
| mustard_bottle | 91.9 | 96.3 | 97.1 | 98.7 | 97.3 | 98.4 | 97.6 | 99.0 |
| tuna_fish_can | 91.1 | 95.3 | 94.1 | 97.1 | 96.9 | 98.5 | 95.8 | 97.8 |
| pudding_box | 85.8 | 92.0 | 80.4 | 90.5 | 97.8 | 98.5 | 76.3 | 89.4 |
| gelatin_box | 96.3 | 97.5 | 96.8 | 99.1 | 97.7 | 98.5 | 96.2 | 99.0 |
| potted_meat_can | 68.7 | 77.9 | 95.4 | 98.1 | 95.1 | 97.7 | 95.6 | 98.2 |
| banana | 74.2 | 86.9 | 92.8 | 98.2 | 96.4 | 98.4 | 95.1 | 98.4 |
| pitcher_base | 86.8 | 94.2 | 97.2 | 98.9 | 96.7 | 98.0 | 97.6 | 99.0 |
| bleach_cleanser | 86.0 | 93.0 | 91.6 | 97.0 | 95.5 | 97.8 | 93.6 | 97.7 |
| bowl | 25.5 | 94.2 | 84.1 | 97.9 | 95.2 | 97.6 | 84.5 | 98.0 |
| mug | 90.9 | 97.1 | 95.6 | 98.4 | 95.6 | 97.9 | 96.0 | 98.6 |
| power_drill | 93.9 | 96.1 | 96.8 | 98.7 | 96.9 | 98.2 | 96.8 | 98.6 |
| wood_block | 20.1 | 89.1 | 91.7 | 96.7 | 93.2 | 97.0 | 94.8 | 97.7 |
| scissors | 76.1 | 85.6 | 94.9 | 97.6 | 94.8 | 97.5 | 93.6 | 97.1 |
| large_marker | 92.0 | 97.1 | 94.0 | 98.1 | 96.9 | 98.6 | 92.5 | 97.8 |
| large_clamp | 48.5 | 94.8 | 93.8 | 97.8 | 93.6 | 97.3 | 94.7 | 98.0 |
| extra_large_clamp | 40.3 | 90.1 | 91.8 | 97.0 | 94.4 | 97.5 | 94.4 | 97.9 |
| foam_brick | 81.1 | 95.7 | 94.0 | 97.9 | 97.9 | 98.6 | 95.0 | 98.0 |
| **All frames** | 80.8 | 93.3 | 94.3 | 97.9 | **96.0** | 97.9 | 95.0 | **98.1** |
| **YCBInEOAT** | | | | | | | | |
| cracker_box | 81.72 | 89.76 | 91.58 | 95.63 | 91.32 | 95.10 | 92.25 | 95.47 |
| bleach_cleanser | 70.79 | 86.84 | 92.06 | 96.07 | 91.45 | 95.96 | 92.32 | 96.17 |
| sugar_box | 88.64 | 93.44 | 93.32 | 96.49 | 94.14 | 96.67 | 94.06 | 96.77 |
| tomato_soup_can | 91.18 | 96.16 | 79.83 | 95.71 | 91.71 | 96.58 | 90.69 | 96.60 |
| mustard_bottle | 92.20 | 96.47 | 95.53 | 97.81 | 96.34 | 97.89 | 95.85 | 97.92 |
| **All frames** | 85.54 | 92.68 | 90.79 | 96.35 | 93.09 | 96.42 | **93.14** | **96.57** |
| FPS [Hz] | 21.7 | | **111.0** | | 41.3 | | 57.6 | |

TABLE III: Ablation on tracking settings on YCBInEOAT.

| **Settings** | **ADD** | **ADD-S** |
|---|---|---|
| SIFT, 1 KF, reprojection error residual | 90.79 | 96.35 |
| + 3D distance residual | 91.53 | 96.36 |
| + SuperPoint instead of SIFT | 92.60 | 96.50 |
| + Increase to 9 keyframes | 93.00 | 96.48 |
| + Enable KF store with 10 keyframes | **93.14** | **96.57** |

with a relative 2.6% gain in the ADD score. This highlights the improvement in rotational tracking accuracy, which is important for robotic manipulation tasks.

### E. Limitations

Since our recovery procedure is based on texture matching with previously seen views of the target object, its effectiveness is limited by the number of view directions observed since the tracking began. This limitation could be mitigated by pre-filling the keyframe store with pre-rendered views of the object if a textured model is available, or with real images taken from a variety of orientations.

## V. Conclusion

In this paper, we presented a novel approach to model-based unseen object pose tracking using RGB-D data. Our proposed tracking reliability check and failure recovery procedure allows the system to detect and recover from tracking failure caused by fast object motions or full occlusions, making it by far the most robust RGB-D object pose tracking method in these challenging scenarios. Furthermore, our approach takes advantage of advances in efficient deep learning-based keypoint detection and combines them with robust texture matching alignment that uses color and depth

information holistically to improve tracking accuracy. Our experiments demonstrate the accuracy and efficiency of our approach on standard benchmarks and highlight its robustness to fast object motions and full occlusions.

## REFERENCES


[1] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proc. of Robotics: Science and Systems (RSS)*, 2024.

[2] W. Deng, D. Campbell, C. Sun, J. Zhang, S. Kanitkar, M.E. Shaffer, and S. Gould. Pos3R: 6D Pose Estimation for Unseen Objects Made Easy. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[3] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox. PoseRBPF: A Rao-Blackwellized Particle Filter for 6D Object Pose Tracking. *arXiv preprint*, arXiv:1905.09304, 2019.

[4] D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperPoint: Self-Supervised Interest Point Detection and Description. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conf.*, 1988.

[6] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[7] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In *Proc. of the Asian Conf. on Computer Vision (ACCV)*, pages 548–562, 2012.

[8] B. Huang, J. Yu, and S. Jain. EARL: Eye-on-Hand Reinforcement Learner for Dynamic Grasping with Active Pose Estimation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2963–2970, 2023.

[9] W. Kehl, F. Tombari, S. Ilic, and N. Navab. Real-Time 3D Model Tracking in Color and Depth on a Single CPU Core. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[10] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proc. of the Intl. Symp. on Mixed and Augmented Reality (ISMAR)*, 2007.

[11] V. Leroy, Y. Cabon, and J. Revaud. Grounding Image Matching in 3D with MASt3R. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 71–91, 2024.

[12] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. DeepIM: Deep Iterative Matching for 6D Pose Estimation. *arXiv preprint*, arXiv:1804.00175, 2018.

[13] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Intl. Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[14] V.N. Nguyen, C. Forster, S. Shkodrani, V. Lepetit, B. Tekin, C. Keskin, and T. Hodan. GoTrack: Generic 6DoF Object Pose Refinement and Tracking. *arXiv preprint*, arXiv:2506.07155, 2025.

[15] V.N. Nguyen, T. Groueix, G. Ponimatkin, V. Lepetit, and T. Hodan. CNOS: A Strong Baseline for CAD-based Novel Object Segmentation. In *Proc. of the Intl. Conf. on Computer Vision Workshops*, 2023.

[16] M. Oquab, T. Darcet, T. Moutakanni, H.V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. ElNouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.Y. Huang, S.W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jégou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: Learning Robust Visual Features without Supervision. *Trans. on Machine Learning Research (TMLR)*, 2024.

[17] E.P. Ornek, Y. Labbe, B. Tekin, L. Ma, C. Keskin, C. Forster, and T. Hodan. FoundPose: Unseen Object Pose Estimation with Foundation Features. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 163–182, 2024.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2019.

[19] V.A. Prisacariu and I.D. Reid. PWP3D: Real-Time Segmentation and Tracking of 3D Objects. *Intl. Journal of Computer Vision (IJCV)*, 98(3):335–354, 2012.

[20] K. Rana, J. Abou-Chakra, S. Garg, R. Lee, I. Reid, and N. Sünderhauf. Affordance-Centric Policy Learning: Sample Efficient and Generalisable Robot Policy Learning using Affordance-Centric Task Frames. *arXiv preprint*, arXiv:2410.12124, 2024.

[21] C.Y. Ren, V.A. Prisacariu, O. Kähler, I.D. Reid, and D.W. Murray. Real-Time Tracking of Single and Multiple Objects from Depth-Colour Imagery Using 3D Signed Distance Functions. *Intl. Journal of Computer Vision (IJCV)*, 124(1):80–95, 2017.

[22] L. Shaikewitz, S. Ubellacker, and L. Carlone. A certifiable algorithm for simultaneous shape estimation and object tracking. *IEEE Robotics and Automation Letters (RA-L)*, 9(12):11873–11880, 2024.

[23] K. Shankar, M. Tjersland, J. Ma, K. Stone, and M. Bajracharya. A Learned Stereo Depth System for Robotic Manipulation in Homes. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):2305–2312, 2022.

[24] J. Shi and C. Tomasi. Good Features to Track. Technical report, Cornell University, 1993.

[25] M. Stoiber, M. Elsayed, A.E. Reichert, F. Steidle, D. Lee, and R. Triebel. Fusing Visual Appearance and Geometry for Multi-Modality 6DoF Object Tracking. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2023.

[26] M. Stoiber, M. Pfanne, K.H. Strobl, R. Triebel, and A. Albu-Schäffer. SRT3D: A Sparse Region-Based 3D Object Tracking Approach for the Real World. *Intl. Journal of Computer Vision (IJCV)*, 130(4):1008–1030, 2022.

[27] M. Stoiber, M. Sundermeyer, and R. Triebel. Iterative Corresponding Geometry: Fusing Region and Depth for Highly Efficient 3D Tracking of Textureless Objects. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[28] H. Tjaden, U. Schwanecke, and E. Schömer. Real-Time Monocular Pose Estimation of 3D Objects using Temporally Consistent Local Color Histograms. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2017.

[29] L. Vacchetti, V. Lepetit, M. Ponder, G. Papagiannakis, P. Fua, D. Thalmann, and N. Magnenat-Thalmann. A Stable Real-Time AR Framework for Training and Planning in Industrial Environments. In *Virtual and Augmented Reality Applications in Manufacturing*, pages 129–146. 2004.

[30] C. Wang, R. Martín-Martín, D. Xu, J. Lv, C. Lu, F.F. Li, S. Savarese, and Y. Zhu. 6-PACK: Category-level 6D Pose Tracker with Anchor-Based Keypoints. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 10059–10066, 2020.

[31] B. Wen and K. Bekris. BundleTrack: 6D Pose Tracking for Novel Objects without Instance or Category-Level 3D Models. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 8067–8074, 2021.

[32] B. Wen, C. Mitash, B. Ren, and K.E. Bekris. se(3)-TrackNet: Data-driven 6D Pose Tracking by Calibrating Image Residuals in Synthetic Domains. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[33] B. Wen, J. Tremblay, V. Blukis, S. Tyree, T. Müller, A. Evans, D. Fox, J. Kautz, and S. Birchfield. BundleSDF: Neural 6-DoF Tracking and 3D Reconstruction of Unknown Objects. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[34] B. Wen, W. Yang, J. Kautz, and S. Birchfield. FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 17868–17879, 2024.

[35] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[36] H. Yang, J. Shi, and L. Carlone. TEASER: Fast and Certifiable Point Cloud Registration. *IEEE Trans. on Robotics (TRO)*, 37(2):314–333, 2020.

[37] L. Zhong and L. Zhang. A Robust Monocular 3D Object Tracking Method Combining Statistical and Photometric Constraints. *Intl. Journal of Computer Vision (IJCV)*, 127(8):973–992, 2019.

[38] Y. Zhou and K. Yang. Exploring TensorRT to Improve Real-Time Inference for Deep Learning. In *Proc. of the Intl. Conf. on High Performance Computing & Communications*, pages 2011–2018, 2022.