arXiv:2506.07155v1 [cs.CV] 8 Jun 2025

GoTrack: Generic 6DoF Object Pose Refinement and Tracking

Van Nguyen Nguyen[1*] Christian Forster[2] Sindi Shkodrani[2] Vincent Lepetit[1]
Bugra Tekin[2] Cem Keskin[2] Tomas Hodan[2]

[1]LIGM, École des Ponts [2]Meta Reality Labs

Abstract

We introduce GoTrack, an efficient and accurate CAD-based method for 6DoF object pose refinement and tracking, which can handle diverse objects without any object-specific training. Unlike existing tracking methods that rely solely on an analysis-by-synthesis approach for model-to-frame registration, GoTrack additionally integrates frame-to-frame registration, which saves compute and stabilizes tracking. Both types of registration are realized by optical flow estimation. The model-to-frame registration is noticeably simpler than in existing methods, relying only on standard neural network blocks (a transformer is trained on top of DINOv2) and producing reliable pose confidence scores without a scoring network. For the frame-to-frame registration, which is an easier problem as consecutive video frames are typically nearly identical, we employ a light off-the-shelf optical flow model. We demonstrate that GoTrack can be seamlessly combined with existing coarse pose estimation methods to create a minimal pipeline that reaches state-of-the-art RGB-only results on standard benchmarks for 6DoF object pose estimation and tracking. Our source code and trained models are publicly available at https://github.com/facebookresearch/gotrack.

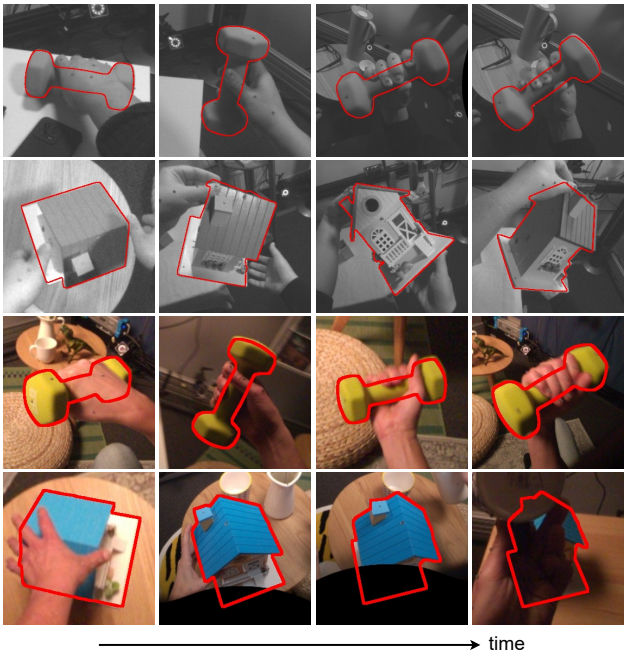

Figure 1. Given a CAD model of the target object, our method can reliably track the 6DoF object pose in challenging conditions such as dynamic scenes with hand-object interactions. The method is applicable to diverse objects and image types, without requiring any domain-specific training. Each row of this figure shows contours of the tracked model in videos from the HOT3D dataset [3] captured by Quest 3 [43] and Project Aria [15] headsets.

1. Introduction

Vision-based 6DoF object pose estimation, which is an important task for robotics and AR/VR, has been significantly improved over the past decade [21, 32, 34, 35, 37, 40, 44, 48, 49, 51, 54, 66, 71, 81]. Yet most methods are designed for single-image input, whereas real-world applications often demand 6DoF object pose tracking, which involves estimating the object pose across consecutive frames.

Classical 6DoF object tracking methods rely on keypoints [52, 56, 57, 61, 69], edges [6, 13, 19, 58, 62], or direct optimization [4, 7, 41, 59, 67]. More recent learning-based tracking methods [8, 17, 42, 72, 75] demonstrate remarkable performance but depend on RGB-D input images [17, 75] or require extensive training datasets specific to each target object [8, 42, 72], which limits their applicability. A problem that is closely related to pose tracking is pose refinement. In both problems, the input is an image, camera intrinsics, and an initial object pose, and the output is a refined object pose. Recent methods [35, 44, 77] for 6DoF object pose refinement rely on an analysis-by-synthesis approach, typically rendering a 3D mesh model in the initial pose and predicting a relative pose w.r.t. the input frame. The pose is predicted either by direct regression [35, 77] or via 2D-

*Work done during Nguyen's internship at Meta.

3D correspondences established between the input image and an RGB-D rendering of the object model [44]. In both cases, the methods need to solve the model-to-frame registration, which is not a trivial problem due to the synthetic-to-real domain gap and requires a relatively heavy network.

In simultaneous localization and mapping (SLAM) [14], the pose between a 3D scene model and a camera is continuously estimated from video frames via two types of registration: model-to-frame and frame-to-frame. The primary role of model-to-frame registration is to correct drift accumulated over time by re-aligning the current frame with the model, ensuring global consistency. On the other hand, frame-to-frame registration focuses on providing smooth, incremental motion estimation as the camera moves through the environment, allowing SLAM methods to efficiently compute the relative transformation between frames. While the interplay of the two registration types have proven effective for SLAM, the 6DoF object tracking field has been exclusively focused on the model-to-frame registration.

In this paper, we introduce GoTrack, a method for generic 6DoF object tracking and pose refinement. The method adapts an analysis-by-synthesis approach for model-to-frame registration, similarly to recent model-based pose refinement methods, while drawing inspiration from SLAM and introducing frame-to-frame registration to reduce computational cost when tracking. Both registration types are realized in GoTrack by optical flow estimation.

For the model-to-frame registration, we train a transformer decoder on top of DINOv2 to predict (1) an optical flow field between a synthetic object template and the input image, and (2) a mask of template pixels that are visible in the input image. From these predictions we construct 2D-3D correspondences by linking visible pixels with 3D points in the model coordinate frame, which are calculated from the depth channel of the template. The 6DoF pose is then calculated by the P*n*P-RANSAC algorithm [16, 36].

For the frame-to-frame registration, we build on the observation that this is an easier problem as the differences between consecutive video frames are typically minimal, and solve it with a small off-the-shelf flow estimation network [65]. The frame-to-frame flow is used to propagate 2D-3D correspondences to the next frame, and the more expensive model-to-frame registration is triggered only if the frame-to-frame tracking is lost. Besides saving compute, the use of the frame-to-frame registration is shown to yield more stable tracking results, especially under occlusion.

Additionally, we show how the proposed method for model-to-frame registration can be seamlessly combined with a BoW-based template retrieval from FoundPose [51] to create an efficient and accurate end-to-end pose estimation pipeline. We demonstrate that the proposed method reaches state-of-the-art results on the standard 6DoF object pose refinement [28] and tracking [68, 79] benchmarks.

In summary, we make the following contributions:

1. *A flow-based 6DoF pose refiner* that is simpler than existing methods, can correct large pose deviations, produces reliable pose confidence score without requiring a scoring network, and reaches state-of-the-art RGB-only results on both 6DoF object pose estimation and tracking benchmarks.
2. *A seamless 6DoF pose estimation pipeline* that swiftly generates coarse poses by template retrieval and optimizes them by the flow-based refiner. The pipeline is easy to implement as both stages rely on DINOv2.
3. *A robust 6DoF pose tracking pipeline* that extends the refiner with frame-to-frame optical flow estimation, which increases tracking efficiency, reduces jitter, and improves robustness to occlusion.

2. Related work

In this section, we provide an overview of existing methods for the closely related problems of pose estimation, refinement and tracking of unseen objects.

Object pose estimation and refinement. Recent works on 6DoF object pose estimation have introduced diverse benchmarks [5, 10, 12, 22, 26, 31, 64, 80] which have powered many deep learning-based methods [29, 32, 34, 37, 38, 40, 53, 54, 66, 81]. As shown in the report of the BOP Challenge 2023 [28], the majority of state-of-the-art methods for 6DoF pose estimation rely on a three-stage pipeline: (1) 2D detection/segmentation, (2) coarse pose estimation, and (3) pose refinement. The top 16 methods for the seen object task and top 13 methods for the unseen object task rely on this pipeline. However, increased accuracy by the third refinement stage usually comes at a significant cost, *e.g.*, the coarse version of GenFlow [44], the winner of BOP 2023, takes 3.8 seconds per image with around 6 objects on average. In contrast, methods based on rendered templates and estimating coarse poses via nearest neighbor search of frozen features of DINOv2 [50], such as ZS6D [2], and FoundPose [51], have shown promising results while being much faster. Motivated by these results, we build our pose refiner on top of frozen DINOv2 [50], which also enables a seamless integration with the recent coarse pose methods.

The refinement stage, which is crucial for achieving state-of-the-art accuracy, is usually realized by an iterative render-and-compare approach with direct regression (CosyPose [34] for seen objects, MegaPose [35] and FoundationPose [77] for unseen objects), or with 2D-3D correspondences (GenFlow [44]). A common property of these methods is that they require a pose ranking/selection step, which requires an extra network that takes as input the test image and renderings of the object in estimated poses, and outputs pose scores. In contrast, our approach produces a reliable pose score without requiring an additional scoring network.

Object pose tracking. Classical object tracking methods can be categorized into keypoint-based [52, 56, 57, 61, 69], edge-based [6, 13, 19, 58, 62], and direct optimization [4, 7, 41, 59, 67]. While keypoints and direct optimization are not suitable for texture-less objects, edge-based methods typically struggle with background clutter and texture. To overcome these issues, learning-based methods [8, 17, 42, 72, 75] have been proposed. However, they either depend on RGB-D input images [17, 75] or require extensive training data specific to each target object [8, 42, 72]. To address these issues, Nguyen et al. [46] proposed a method for tracking 6DoF pose of unseen objects. While their approach only demonstrates results in simple scenarios or simple motions, we focus on more challenging conditions where objects can be heavily occluded or manipulated by hands. Another line of work includes category-level methods [39, 70], which focus on unseen instances belonging to training categories. Recently, several methods for instance-level tracking [8, 18, 37, 75], model-based [30, 63, 78], and model-free tracking of unseen objects [74, 76] have been proposed. However, these methods either focus on seen objects or use depth images, which limits their applicability.

Object pose refinement for tracking of unseen objects. The most similar to our approach are methods introduced in [35, 37, 44, 77]. A prominent example is DeepIM [37], which introduced a deep-learning approach for iterative render-and-compare pose refinement. While their method shows generalization to unseen objects, the experiments are limited to synthetic images. MegaPose [35] has further extended this approach and shown strong generalization to unseen objects in real images by training on large-scale datasets. In the BOP Challenge 2023 [28], GenFlow [44] won the award by introducing a RAFT [65]-based architecture to estimate the flow from the template to real input images. In this work, we also introduce a flow-based method for unseen object pose refinement. However, our approach does not require costly shape constraints, differentiable P*n*P, specialized neural architecture, nor a pose scoring stage to reach state-of-the-art results. Moreover, all of the existing methods focus solely on model-to-frame registration. When applied to tracking, they transition to the next frame solely via pose parameters, discarding pixel-level information about the established registration. In this work, we show that propagating 2D-3D correspondences via frame-to-frame flow increases efficiency and reduces jitter.

3. Method

In this section we first propose a 6DoF pose refinement method for unseen objects (Sec. 3.1), and then show how this method can support efficient pipelines for estimating accurate object pose from a single image (Sec. 3.2) and for tracking the object pose in an image sequence (Sec. 3.3).

3.1. Flow-based object pose refinement

Problem formulation. Given a CAD model of an object, an RGB image with known intrinsics that shows the object in an unknown pose, and a coarse estimate of the object pose, our objective is to refine the pose such as the 2D projection of the model aligns closely with the object's appearance in the image. We assume the object is rigid and represent its pose by a rigid transformation $(\mathbf{R}, \mathbf{t})$, where $\mathbf{R}$ is a 3D rotation and $\mathbf{t}$ is a 3D translation from the model coordinate frame to the camera coordinate frame.

Method overview. We adapt an analysis-by-synthesis approach, similarly to several recent methods for model-based pose refinement [35, 37, 44, 77]. In particular, we predict the refined pose based on (i) an RGB-D template showing the CAD model in the given coarse pose, and (ii) a crop of the object region in the input RGB image (the region is determined by the CAD model and the coarse pose). Instead of directly regressing a relative 6DoF pose between the rendering and the crop (as in [35, 37, 77]), we train a network to predict dense 2D-2D correspondences, represented as an optical flow field, together with a mask of pixels that are visible in both images. The pose is then estimated by lifting the correspondences to 2D-3D using the template depth, and by applying P*n*P-RANSAC [16, 36]. The refinement method can be applied in several iterations for increased accuracy.

Predicting 2D outputs (flow and visibility mask) instead of the relative 6DoF pose makes the network easier to train, interpretable and versatile – we show how to use the predicted correspondences to calculate a reliable and controllable pose score and how to naturally propagate them via frame-to-frame flow in case of tracking. The correspondence-based approach could also support 2D tracking (just by turning off the 3D reasoning part).

Predicting 2D-3D correspondences. Let $\mathbf{I}_c$ be a perspective crop of the input RGB image around a provided 2D bounding box of the object. The crop is obtained as in [51] by warping the image to a virtual pinhole camera whose optical axis passes through the center of the 2D bounding box. Then let $\mathbf{I}_t$ be an RGB-D template that is rendered using a pinhole camera and shows the object in an initial pose $(\mathbf{R}_0, \mathbf{t}_0)$, let $\mathbf{M}_t$ be a binary mask of the object silhouette in $\mathbf{I}_t$, and let O be a set of 3D points representing the surface of the object model. $\mathbf{I}_t$ and $\mathbf{I}_c$ have the same resolution.

Given the crop $\mathbf{I}_c$ and the template $\mathbf{I}_t$, we aim to establish correspondences between their pixels. We deem pixels $\mathbf{u}_t \in \mathbf{I}_t$ and $\mathbf{u}_c \in \mathbf{I}_c$ as corresponding if they are aligned with 2D projections of the same 3D point $\mathbf{x} \in O$: $\mathbf{u}_t = \pi_t(\mathbf{R}_0\mathbf{x} + \mathbf{t}_0)$ and $\mathbf{u}_c = \pi_c(\bar{\mathbf{R}}\mathbf{x} + \bar{\mathbf{t}})$, where $\pi\colon \mathbb{R}^3 \mapsto \mathbb{R}^2$ is the 2D projection operator, and $(\bar{\mathbf{R}}, \bar{\mathbf{t}})$ is the GT object pose.

To establish the correspondences, we train a neural network to predict two outputs for each pixel $\mathbf{u}_t \in \mathbf{M}_t$: a like-

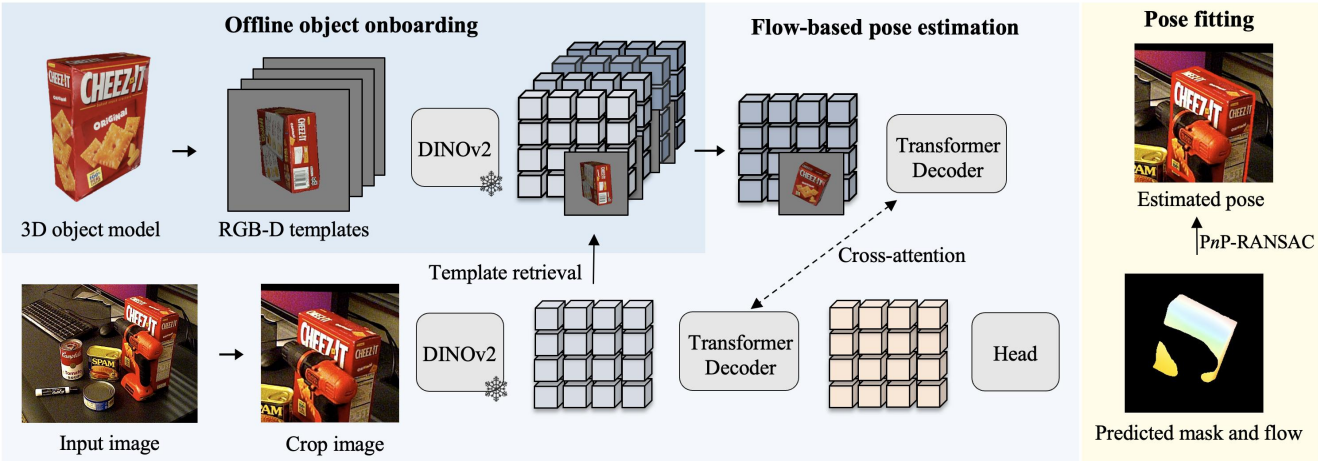

Figure 2. **Overview of the proposed efficient pipeline for 6DoF pose estimation.** Each object is first onboarded by rendering RGB-D templates of its 3D model, and by extracting DINOv2 [50] feature maps from RGB channels of the templates. At inference, we first crop the input image around a 2D object detection (from CNOS [47]) and retrieve a template that shows the object in a similar pose as in the crop by the BoW-based approach from [51]. Then we use a transformer decoder [73] with a DPT head [55] to predict (1) 2D-2D correspondences between the template and the input, and (2) a mask of template pixels that are visible in the input. Next we lift the 2D-2D correspondences established at visible template pixels to 2D-3D using the depth channel of the template, and estimate the pose by P*n*P-RANSAC.

lihood $v_{t\to c}(\mathbf{u}_t)$ that the corresponding pixel $\mathbf{u}_c \in \mathbf{I}_c$ is visible, and a 2D flow vector $\mathbf{f}_{t\to c}(\mathbf{u}_t)$ such that $\mathbf{u}_t + \mathbf{f}_{t\to c}(\mathbf{u}_t) = \mathbf{u}_c$. Given the network predictions, we collect a set of weighted 2D-2D correspondences $D = \{(\mathbf{u}_c, \mathbf{u}_t, w)\}$ by linking pixels $\mathbf{u}_u$ and $\mathbf{u}_t$ if they are related by $\mathbf{f}_{t\to c}(\mathbf{u}_t)$ and if $v_{t\to c}(\mathbf{u}_t)$ is above a threshold τ_v. The weight w is defined by $v_{t\to c}(\mathbf{u}_t)$. Finally, we convert each linked template pixel $\mathbf{u}_t$ to a 3D point $\mathbf{x} \in O$ using the template depth and known camera intrinsics, to obtain a set of weighted 2D-3D correspondences $C = \{(\mathbf{u}_c, \mathbf{x}, w)\}$.

Pose fitting. The refined pose $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$ is estimated from the established correspondences C by solving the Perspective-*n*-Point (P*n*P) problem. As in [51], we solve this problem by the EP*n*P algorithm [36] combined with the RANSAC fitting scheme [16] for robustness. In this scheme, P*n*P is solved repeatedly on a randomly sampled minimal set of 4 correspondences, and the output is defined by the pose hypothesis with the highest number of inlier correspondences $C' \subseteq C$, for which the 2D re-projection error [36] is below a threshold τ_r. The final pose is then refined from all inliers by the Levenberg-Marquardt optimization [45].

Quality of pose estimates. We define the quality of an estimated pose $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$ as: $q = s'/s$, where s' is the sum of weights w of inlier correspondences C', and s is the sum of weights w of all correspondences C. As shown in Sec. 4, this simple quality measure is surprisingly effective for selecting the best pose estimate in a multiple-hypotheses refinement setup. Note that existing methods [35,44,77] need a special pose scoring network for this purpose.

Network architecture and training. We rely on a frozen DINOv2 [50] backbone that we use to extract features from the image crop $\mathbf{I}_c$ and the template image $\mathbf{I}_t$. We then pass the features to a two-branch transformer-based decoder from CroCov2 [73] (one branch for the template and one for the crop), and finally obtain the predictions by applying the DPT head [55] to the output of the template branch. The decoder and the head are trained jointly by minimizing the following loss averaged over all pixels $\mathbf{u}_t \in \mathbf{M}_t$:

$$\mathcal{L}(\mathbf{u}_t) = \text{BCE}\big(v_{t\to c}(\mathbf{u}_t),\ \bar{v}_{t\to c}(\mathbf{u}_t)\big) + \bar{v}_{t\to c}(\mathbf{u}_t)\ \big\|\mathbf{f}_{t\to c}(\mathbf{u}_t) - \bar{\mathbf{f}}_{t\to c}(\mathbf{u}_t)\big\|_1,$$

where BCE is the binary cross entropy loss between the predicted visibility $v_{t\to c}(\mathbf{u}_t)$ and the binary ground-truth visibility $\bar{v}_{t\to c}(\mathbf{u}_t)$. As in RAFT [65], the optical flow prediction is supervised by minimizing the L1 distance between the predicted flow vector $\mathbf{f}_{t\to c}(\mathbf{u}_t)$ and the ground-truth flow vector $\bar{\mathbf{f}}_{t\to c}(\mathbf{u}_t)$. Note that the flow prediction is supervised only at template pixels $\mathbf{u}_t \in \mathbf{M}_t$ for which the corresponding crop pixel $\mathbf{u}_c \in \mathbf{M}_c$ is visible, *i.e.*, when $\bar{v}_{t\to c}(\mathbf{u}_t) = 1$. The ground-truth visibility value $\mathbf{u}_c \in \mathbf{M}_c$ is defined by $\mathbf{M}_c(\mathbf{u}_t + \mathbf{f}_{t\to c}(\mathbf{u}_t))$, where $\mathbf{M}_c$ is a modal mask defined in the crop ($\mathbf{M}_c$ is required only at training).

We train the network on the MegaPose-GSO dataset [35] that offers 1M PBR images rendered with BlenderProc [9], shows 1000 GSO objects [11] that are annotated with the GT 6DoF poses and 2D segmentation masks. As in [35], we apply heavy image augmentation during training, which avoid overfitting to the synthetic image domain and enables applying the network to real images at inference time. We also randomly perturb the GT poses as in [35].

3.2. Efficient object pose estimation pipeline

Similarly to the state-of-the-art methods for 6DoF pose estimation of unseen objects [28], we adopt a three-stage pipeline: 2D object detection, coarse 6DoF pose estimation, 6DoF pose refinement. While the first stage can be realized by any 2D object detector (*e.g*. CNOS [47]), the key difference of our pipeline is in the other two stages, where we apply efficient template retrieval for coarse pose estimation and our flow-based approach for pose refinement, both based on frozen DINOv2 [50]. This streamlined pipeline delivers competitive accuracy while being faster than existing methods (Sec. 4). Fig. 2 illustrates the pipeline.

Object onboarding. First, in an offline onboarding stage, we pre-render a set of RGB-D templates (800 in our experiments), similarly to [51]. Given a texture-mapped 3D object model, we render templates showing the model under orientations sampled to uniformly cover the SO(3) group of 3D rotations [1], and the model is rendered using a standard rasterization technique [60] with a gray background and fixed lighting. We also pre-extract DINOv2 feature maps from RGB channels of the templates.

Coarse pose estimation by template retrieval. Given a 2D object detection (*e.g*., from CNOS [47]), we generate a coarse object pose with the fast template retrieval approach from FoundPose [51]. In Sec. 4, we show that poses associated with the retrieved templates are sufficient for our flow-based refiner to converge to an accurate pose.

Seamless integration. The template retrieval approach from [51] leverages Bag-of-Words (BoW) descriptors that are computed from patch features from frozen DINOv2. Since our refiner utilizes the same backbone, the feature extraction process needs to be performed only once per image crop. The extracted DINOv2 features can then be used both to retrieve templates and to refine the poses associated. Besides sharing the same backbone, the two stages in fact share the exact same template-based object representation.

3.3. Extension to object pose tracking

Limitation of tracking by refinement. The proposed refiner can be directly applied to track the 6DoF object pose in an image sequence. This can be achieved as in [35, 77] by taking the pose from the previous frame as an initial pose for the next frame. Although straightforward, this approach tends to suffers from jitter, *i.e*., rapid fluctuations or inconsistencies, often resulting in a shaky or unstable appearance in the tracking output. We argue that this limitation arises because predictions between consecutive frames are connected solely through pose parameters. Information about the pixel-level model-to-image registration from the previous frame is discarded. Consequently, the registration process, a core challenge in all analysis-by-synthesis refinement methods, must largely restart in each new frame. We address this limitation by introducing frame-to-frame consistency, which we introduce next.

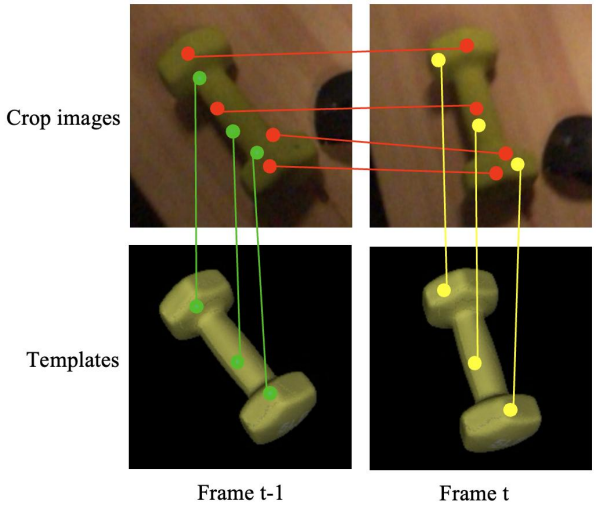

Figure 3. **Frame-to-frame consistency for tracking.** Besides the model-to-frame registration (between a synthetic template and the real input image), which is the main building block of existing 6DoF object tracking methods, we introduce frame-to-frame registration, which reduces jitter and increases efficiency. As the difference between consecutive video frames is typically minimal, estimating frame-to-frame correspondences is a simpler problem and can be solved by a light-weight optical flow network such as RAFT [65]. The model-to-frame correspondences need to be established only in case of a low frame-to-frame tracking confidence.

Frame-to-frame consistency. Inspired by the SLAM literature [14], where the pose between a 3D scene model and a camera is continuously estimated from video frames not only by model-to-frame but also by frame-to-frame registration, we introduce the direct frame-to-frame connection to our pipeline by predicting optical flow between the previous and the current video frames.

Specifically, we obtain a perspective crop of the current frame j by constructing a virtual camera looking at the object model in the current pose estimate, and use RAFT [65] to predict optical flow between the previous and the current crop. We then apply this flow to the 2D coordinates of the inlier 2D-3D correspondences C'_i established in the previous frame i to obtain a set of propagated inliers $C'_{i\to j}$, and re-estimate the pose by the EP*n*P algorithm from $C'_{i\to j}$. If the inlier ratio w.r.t. the re-estimated pose is above a threshold τ_i (typically 0.8), we use the re-estimated pose as the final pose estimate at frame j and proceed to the next frame. Otherwise, if the inlier ratio is below the threshold, we predict the model-to-frame flow (as described earlier) to obtain a new set of 2D-3D correspondences C_j. In this case, to encourage temporal pose smoothness, we combine sets $C'_{i\to j}$

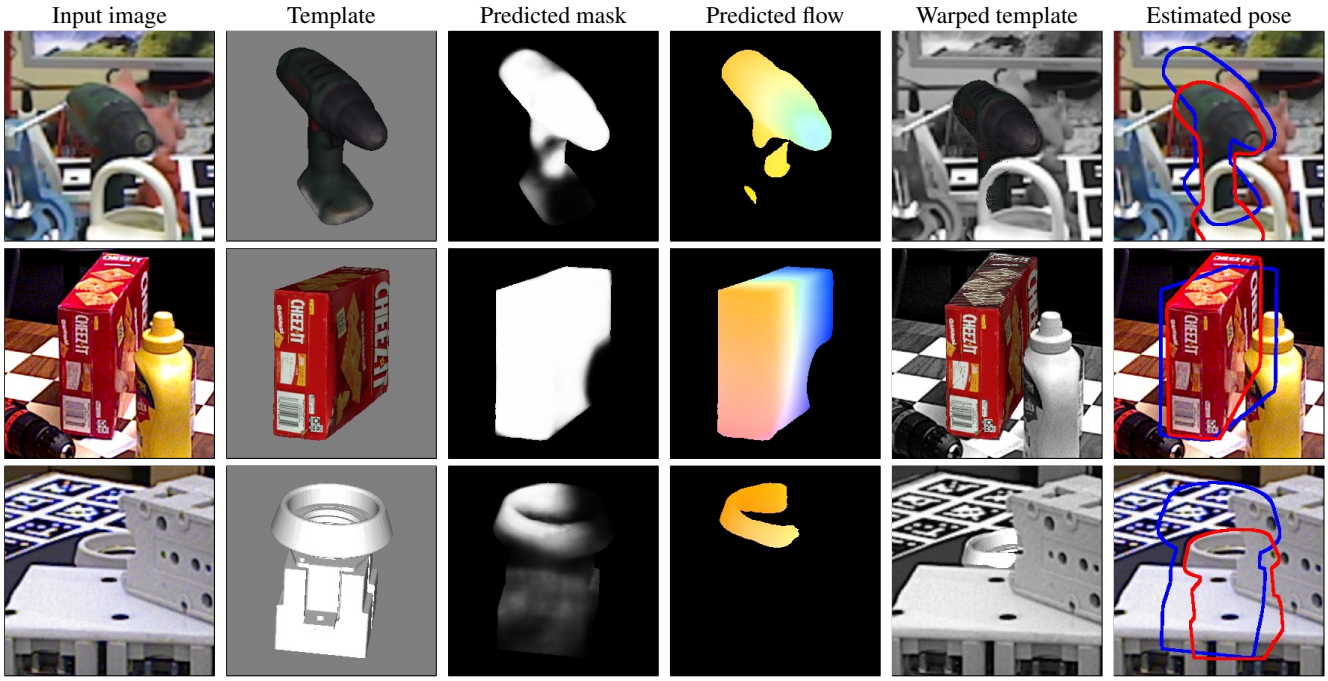

Figure 4. **Example results of the GoTrack refiner on LM-O [5], YCB-V [79], and T-LESS [23].** The input image is shown in the first column, and the template retrieved using the BoW-based approach [51] in the second. The third and fourth columns show the predictions of our network, which can then be used to remap pixels from the template to the input image as shown in the fifth column. The last column presents the final pose estimated by P*n*P-RANSAC from 2D-3D correspondences (the contour of the object model in the initial pose is shown in blue, and in the estimated pose in red). As shown in the third column, our method can reliably predict which part of the object is visible, despite never seeing the object during training.

and C_j by taking all correspondences from the first set and randomly selecting up to $r|C'_{i\to j}|$ correspondences from the second, *i.e.*, such as the ratio of correspondences selected from the two sets is up to r. The pose is then estimated from the combined set by EP*n*P, and inlier correspondences from the combined set are propagated to the next frame.

The parameter r, which controls the ratio of correspondences selected from the previous and the current frame, can be effectively used to control the trade-off between jitter and the risk of drift (*i.e.*, the risk of deviating from the actual object pose). For lower values of r the tracking tends to be smoother but may suffer from drifting, while for higher values of r the tracking may be jittery.

4. Experiments

In this section, we describe the experimental setup for 6DoF object pose refinement (Sec. 4.1) and for 6DoF object pose tracking (Sec. 4.2), and compare GoTrack with the state-of-the-art RGB-only methods for these tasks. Specifically, we compare GoTrack with refinement methods MegaPose [35] and GenFlow [44] on BOP datasets [28], with tracking methods LDT3D [67], SRT3D [62] and DeepAC [72] on the RBOT dataset [68], and tracking methods PoseCNN [79] and PoseRBPF [8] on the YCB-V dataset [79]. Among the tracking methods, only LDT3D and SRT3D operate on unseen objects while DeepAC, PoseCNN, and PoseRBPF are trained on the test objects.

Implementation details. The evaluated version of the proposed refinement method used 800 templates (with approximately 25° angle between depicted object orientations), the template and crop resolution of 280×280 px, DINOv2 [50] with the ViT-S architecture, the transformer decoder from CroCov2 [73] with 12 blocks, the default DPT head [55], the visibility threshold τ_v set to 0.3, up to 400 iterations of P*n*P-RANSAC, and the re-projection threshold τ_r set to 4 px. For the pose refinement experiments, we use 5 refinement iterations as previous works [35, 44]. Although our flow-based refiner could rely solely on templates pre-rendered in the onboarding phase (at each refinement iteration, we could use the template that best matches the orientation of the current pose estimate), we found that this approach does not perform as well as online rendering, where the 3D model is rendered in the current pose estimate and therefore does not suffer from the pose quantization error. Online rendering is used also in the prior works [35,44]. For

#	Method	LM-O		T-LESS		TUD-L		IC-BIN		ITODD		HB		YCB-V		Average↑		Time↓	Nets↓
		AR_{MSPD}	AR	AR_{MSPD}	AR	AR_{MSPD}	AR	AR_{MSPD}	AR	AR_{MSPD}	AR	AR_{MSPD}	AR	AR_{MSPD}	AR	AR_{MSPD}	AR		
Coarse pose estimation:																			
1	FoundPose [51]	**60.7**	**39.6**	**48.9**	**33.8**	**67.0**	**46.7**	33.9	**23.9**	**37.0**	**20.4**	**65.0**	**50.8**	**59.6**	**45.2**	**52.2**	**37.2**	1.7 s	1
2	GigaPose [48]	51.0	29.6	47.9	26.4	51.7	30.0	**34.8**	22.3	31.4	17.5	52.5	34.1	52.5	27.8	45.4	26.8	0.4 s	1
3	BoW retrieval*	25.9	11.6	28.8	15.4	25.3	13.7	17.4	10.1	31.6	17.4	55.1	42.4	15.2	11.6	28.5	17.5	**0.3** s	1
Refinement using a single hypothesis from FoundPose [51]:																			
4	GoTrack	**70.4**	**56.5**	**57.3**	50.4	**81.5**	**67.2**	**50.7**	**43.2**	**52.8**	**39.3**	**76.3**	**72.2**	75.8	63.1	**66.4**	**56.0**	**2.5** s	1
5	MegaPose [35]	67.7	55.4	55.6	**51.0**	78.4	63.3	47.4	43.0	45.2	34.6	73.6	69.5	**76.0**	**66.1**	63.4	54.7	4.4 s	2
Refinement using a single hypothesis from GigaPose [48]:																			
6	GoTrack	**72.6**	58.4	**61.9**	**54.9**	**73.9**	**61.3**	**52.5**	45.3	**50.0**	39.5	**77.0**	**72.9**	**73.8**	62.0	**66.0**	56.3	**1.3** s	2
7	GenFlow [44]	71.1	**59.5**	60.1	55.0	72.4	60.7	52.1	**47.8**	49.5	**41.3**	74.7	72.2	73.2	60.8	64.7	**56.8**	2.2 s	2
8	MegaPose [35]	68.0	55.7	59.2	54.1	71.9	58.0	49.4	45.0	45.4	37.6	73.1	69.3	73.4	**63.2**	64.2	54.7	2.3 s	2
Refinement using BoW retrieval for coarse pose estimation (from row 3):																			
9	GoTrack	70.4	56.9	56.9	50.4	79.2	65.2	49.7	42.4	52.6	40.2	72.5	69.5	75.2	62.9	65.2	55.4	0.8 s	1
Pose refinement using 5 hypotheses from GigaPose [48]:																			
12	GoTrack	**78.1**	62.7	**66.3**	**59.5**	**80.8**	**66.8**	54.4	46.6	**56.3**	43.0	**81.9**	**77.1**	**80.6**	**68.4**	**71.2**	**60.6**	**3.0** s	2
13	GenFlow [44]	75.3	**63.1**	63.7	58.2	79.4	66.4	**55.1**	**49.8**	55.1	**45.3**	78.3	75.6	78.8	65.2	69.4	60.5	10.6 s	3
14	MegaPose [35]	73.6	59.8	62.0	56.5	77.9	63.1	52.8	47.3	48.5	39.7	76.5	72.2	77.7	66.1	65.3	57.8	7.7 s	3

Table 1. **Pose refinement performance on the seven core BOP datasets [28].** Reported are accuracy scores, the average time required to estimate poses of all objects in an image (in seconds), and the number of different neural networks required by the pipelines.

the object tracking experiments, the parameter τ_i that triggers the template-to-frame registration was set to 0.8, and the mixing ratio r was set to 2. We trained the flow-based refiner until convergence on the MegaPose-GSO dataset [35] using Adam [33], with DINOv2 weights being frozen.

4.1. 6DoF object pose refinement

Evaluation datasets. We evaluate our method on the seven core BOP datasets [28]: LM-O [5], T-LESS [22], TUD-L [26], IC-BIN [10], ITODD [12], [31], and YCB-V [80]. These datasets include the total of 132 different objects and 19048 annotated test object instances.

Evaluation metrics. We use the BOP evaluation protocol [27], which relies on three metrics: Visible Surface Discrepancy (VSD), Maximum Symmetry-Aware Surface Distance (MSSD), and Maximum Symmetry-Aware Projection Distance (MSPD). The final average recall (AR), is calculated by averaging the individual AR scores of these three metrics across a range of error thresholds.

Baselines. We compare our method with MegaPose [35] and GenFlow [44], which are currently the state-of-the-art methods on the BOP benchmark. The results of MegaPose and GenFlow are sourced from the BOP leaderboard [28].

Results. When starting from coarse poses estimated by FoundPose [51], our refiner outperforms the MegaPose refiner [35] on the majority of BOP datasets, achieving +3.0 AR_{MSPD} and +1.3 AR on average (rows 4 and 5 in Tab. 1). When starting from coarse poses estimated by GigaPose [48], our refiner outperforms MegaPose by similar margins (rows 6 and 8). Compared to GenFlow [44], our refiner achieves +1.3 AR_{MSPD} and -0.5 AR (rows 6 and 7).

In terms of complexity of the model architecture, the simplest from the evaluated methods is the pipeline proposed in Sec. 3.2, which relies on the BoW-based template retrieval for coarse pose estimation followed by our refiner. This pipeline relies on only a single neural network (column "Nets") that is shared by both stages, which makes it faster and easier to deploy on mobile devices. Comparing runtime of other methods is problematic as each was evaluated on a different GPU (we used V100, while MegaPose was evaluated on RTX 2080 and GenFlow on RTX 3090). However, the difference in complexity of the methods is clear. Both GenFlow and MegaPose pipelines rely on different networks for coarse pose estimation and for refinement. In case of multi-hypotheses setup, where multiple coarse poses are considered per object instance and their refined versions need to be at the end ranked to select the final output, the GenFlow and MegaPose pipelines need an additional scoring network, whereas our method uses reliable pose confidence score can be obtained by simply calculating the weighted inlier ratio defined in Sec. 3.1.

4.2. 6DoF object pose tracking

Evaluation datasets. We evaluate our tracking method on two popular datasets: YCB-V [79] and RBOT [68]. Additional tracking results on the recent HOT3D [3] dataset are presented in the appendix.

Evaluation metrics. We use the same standard evaluation metrics as in [8, 62, 67, 72, 79]: Area Under the Curve

	PoseCNN [79]		PoseRBPF [8]		GoTrack		GoTrack+f2f	
Initialization	-		PoseCNN		FoundPose		FoundPose	
Unseen objects	✗		✗		✓		✓	
	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S
002_master_chef_can	50.9	84.0	58.0	77.1	71.0	**86.3**	**71.3**	**86.3**
003_cracker_box	51.7	76.9	76.8	87.0	79.6	89.2	**80.6**	**89.8**
004_sugar_box	68.6	84.3	75.9	87.6	80.4	89.9	**80.5**	**90.0**
005_tomato_soup_can	66.0	80.9	**74.9**	84.5	74.0	**87.2**	71.2	85.8
006_mustard_bottle	79.9	90.2	82.5	91.0	83.7	91.9	**84.4**	**92.2**
007_tuna_fish_can	70.4	87.9	59.0	79.0	81.4	91.7	**82.0**	**91.9**
008_pudding_box	62.9	79.0	57.2	72.1	82.8	90.4	**83.5**	**90.7**
009_gelatin_box	75.2	87.1	**88.8**	**93.1**	83.4	90.9	**84.4**	**91.5**
010_potted_meat_can	59.6	78.5	49.3	62.0	80.5	91.3	**83.5**	**92.8**
011_banana	72.3	85.9	24.8	61.5	53.8	**71.0**	**56.0**	68.7
019_pitcher_base	52.5	76.8	75.3	88.4	79.7	90.0	**79.9**	**90.1**
021_bleach_cleanser	50.5	71.9	54.5	69.3	68.9	83.1	**69.3**	**83.3**
024_bowl	6.5	69.7	**36.1**	**86.0**	35.1	66.2	34.8	62.0
025_mug	57.7	78.0	**70.9**	85.4	70.1	87.1	50.8	**87.4**
035_power_drill	55.1	72.8	70.9	85.0	81.9	91.2	**82.1**	**91.3**
036_wood_block	**31.8**	**65.8**	2.8	33.3	0.7	30.4	5.3	40.9
037_scissors	35.8	56.2	21.7	33.0	58.3	76.5	**59.7**	**77.8**
040_large_marker	58.0	71.4	48.7	59.3	60.0	67.3	**65.0**	**73.9**
051_large_clamp	25.0	49.9	47.3	76.9	64.1	83.1	**64.4**	**83.2**
052_extra_large_clamp	15.8	47.0	26.5	69.5	**83.5**	**93.5**	82.8	93.0
061_foam_brick	40.4	87.8	78.2	89.7	83.0	92.3	**83.3**	**92.4**
Average	53.7	75.9	59.9	77.5	**69.3**	82.9	**69.3**	**83.6**

Table 2. **Tracking performance on the YCB-V dataset [79].** The accuracy is measured by AUC w.r.t. the ADD and ADD-S pose error functions. Despite not being trained on the YCB-V objects, our method outperforms PoseCNN [79] and PoseRBPF [8] which were trained on these specific objects. Adding the frame-to-frame consistency (f2f) makes our method 6X more efficient while slightly improving the accuracy (see text for details).

Method	with reset						w/o reset
	Regular	Dynamic	Noisy	Occlusion	Mean	Resets ↓	Mean
SRT3D [62]	94.2	94.6	81.7	93.2	90.9	6575	19.0
LDT3D [67]	95.2	95.4	83.2	94.9	92.1	6228	18.8
DeepAC [72]*	95.6	95.6	88.0	94.0	93.3	4826	30.3
GoTrack	**97.3**	**96.2**	**94.2**	**95.4**	**95.9**	**3021**	**86.7**

Table 3. **Tracking on the RBOT dataset [68] with/without reset** as in [62, 67, 72]. The accuracy is measured by 5*cm*-5°score and the number of reset. Character * indicates that DeepAC is a supervised method trained on the test objects. GoTrack significantly outperforms existing methods across all settings and metrics.

(AUC) for the ADD and and ADD(-S) pose error functions on YCB-V dataset, and 5*cm*-5°and the number of resets on RBOT dataset. A reset is counted when the tracking method fails, *i.e.*, when the translation error exceeds 5 centimeters or the rotation error exceeds 5 degrees, the method is re-initialized with the ground-truth pose. More details about these metrics are in the appendix.

Baselines. Since no RGB-only methods for tracking of unseen objects have been evaluated on the YCB-V dataset, we compare our method with tracking methods for seen objects (*i.e.*, methods trained on the target objects), PoseCNN [79] and PoseRBPF [8]. PoseCNN is a method for 6DoF object pose estimation from a single image and was evaluated frame by frame. PoseRBPF was initialized at the first frame with a pose estimated by PoseCNN. Since the PoseCNN estimates are not publicly available, we initialized our tracking method with a pose estimate from FoundPose. On the RBOT dataset, we compare our method with contour-based methods DeepAC [72] and SRT3D [62] and the optimization-based method LDT3D [67].

Results on YCB-V [79]. As shown in Tab. 2, even though our method was not trained on the YCB-V objects, it significantly outperforms both PoseCNN [79] and PoseRBPF [8], which were trained specifically for these objects. Adding the frame-to-frame flow makes our method around 6X more efficient while slightly improving the accuracy. The forward pass of the flow-based refiner, which we use for the template-to-frame registration, takes 40 ms while the forward pass of RAFT-Small, which we use for the frame-to-frame registration takes only 6 ms. The template-to-frame flow is triggered only every 50th frame on average on sequences from the YCB-V dataset. Another effect of adding the frame-to-frame flow is a reduced jitter, which can be seen in the supplementary video (App. D).

Results on RBOT [68]. As shown in Tab. 3, GoTrack outperforms LDT3D, SRT3D and DeepAC on all RBOT sequence types while requiring 37–54% less resets. The most noticeable difference is on the noisy sequences where contour detection is less reliable. Furthermore, these methods work well only on objects with distinct contour – they struggle if the same contour shape corresponds to multiple different poses and the actual pose can be estimated only based on texture (*e.g.*, DeepAC achieves only 70.1% 5*cm*-5°accuracy on the textured and cylindrical soda object from RBOT while GoTrack achieves 97.2%). Additionally, Tab. 3 shows that our method significantly outperforms the other methods in tracking without reset (86.7 vs 30.3 AUC).

5. Conclusion

We introduced a method for 6DoF pose refinement of unseen objects, based on predicting template-to-frame optical flow along with a mask of visible template pixels. We showed how our refinement method can be seamlessly combined with a BoW-based template retrieval to create an efficient and accurate object pose estimation pipeline. Additionally, we proposed an extension for 6DoF pose tracking of unseen objects by propagating 2D-3D correspondences using frame-to-frame optical flow. Our approach achieves state-of-the-art results on the standard 6DoF object pose refinement and tracking benchmarks. For future work, we plan to draw further inspiration from SLAM literature and explore extending our approach to a model-free setup.

References

[1] Marc Alexa. Super-fibonacci spirals: Fast, low-discrepancy sampling of so (3). In *CVPR*, 2022. 5

[2] Philipp Ausserlechner, David Haberger, Stefan Thalhammer, Jean-Baptiste Weibel, and Markus Vincze. ZS6D: Zero-Shot 6D Object Pose Estimation Using Vision Transformers. 2023. 2

[3] Prithviraj Banerjee, Sindi Shkodrani, Pierre Moulon, Shreyas Hampali, Fan Zhang, Jade Fountain, Edward Miller, Selen Basol, Richard Newcombe, Robert Wang, et al. Introducing hot3d: An egocentric dataset for 3d hand and object tracking. *arXiv preprint arXiv:2406.09598*, 2024. 1, 7, 12, 13

[4] Selim Benhimane and Ezio Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004. 1, 3

[5] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D Object Pose Estimation Using 3D Object Coordinates. In *ECCV*, 2014. 2, 6, 7, 14

[6] Andrew I. Comport, Eric Marchand, Muriel Pressigout, and Francois Chaumette. Real-time markerless tracking for augmented reality: The virtual visual servoing framework. *IEEE Transactions on Visualization and Computer Graphics*, 2006. 1, 3

[7] Alberto Crivellaro and Vincent Lepetit. Robust 3D tracking with descriptor fields. In *CVPR*, 2014. 1, 3

[8] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. PoseRBPF: A Rao–Blackwellized particle filter for 6-D object pose tracking. *IEEE Transactions on Robotics*, 2021. 1, 3, 6, 7, 8, 12

[9] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Dmitry Olefir, Tomáš Hodaň, Youssef Zidan, Mohamad Elbadrawy, Markus Knauer, Harinandan Katam, and Ahsan Lodhi. BlenderProc: Reducing the reality gap with photorealistic rendering. *RSS Workshops*, 2020. 4

[10] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. Recovering 6D Object Pose and Predicting Next-Best-View in the Crowd. In *CVPR*, 2016. 2, 7, 18

[11] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3D scanned household items. *ICRA*, 2022. 4

[12] Bertram Drost, Markus Ulrich, Paul Bergmann, Philipp Hartinger, and Carsten Steger. Introducing MVTec ITODD – A dataset for 3D object recognition in industry. In *ICCVW*, 2017. 2, 7, 20

[13] Tom Drummond and Roberto Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002. 1, 3

[14] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 2006. 2, 5

[15] Jakob Engel, Kiran Somasundaram, Michael Goesele, Albert Sun, Alexander Gamino, Andrew Turner, Arjang Talattof, Arnie Yuan, Bilal Souti, Brighid Meredith, Cheng Peng, Chris Sweeney, Cole Wilson, Dan Barnes, Daniel DeTone, David Caruso, Derek Valleroy, Dinesh Ginjupalli, Duncan Frost, Edward Miller, Elias Mueggler, Evgeniy Oleinik, Fan Zhang, Guruprasad Somasundaram, Gustavo Solaira, Harry Lanaras, Henry Howard-Jenkins, Huixuan Tang, Hyo Jin Kim, Jaime Rivera, Ji Luo, Jing Dong, Julian Straub, Kevin Bailey, Kevin Eckenhoff, Lingni Ma, Luis Pesqueira, Mark Schwesinger, Maurizio Monge, Nan Yang, Nick Charron, Nikhil Raina, Omkar Parkhi, Peter Borschowa, Pierre Moulon, Prince Gupta, Raul Mur-Artal, Robbie Pennington, Sachin Kulkarni, Sagar Miglani, Santosh Gondi, Saransh Solanki, Sean Diener, Shangyi Cheng, Simon Green, Steve Saarinen, Suvam Patra, Tassos Mourikis, Thomas Whelan, Tripti Singh, Vasileios Balntas, Vijay Baiyya, Wilson Dreewes, Xiaqing Pan, Yang Lou, Yipu Zhao, Yusuf Mansour, Yuyang Zou, Zhaoyang Lv, Zijian Wang, Mingfei Yan, Carl Ren, Renzo De Nardi, and Richard Newcombe. Project Aria: A new tool for egocentric multi-modal AI research, 2023. 1

[16] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981. 2, 3, 4

[17] Mathieu Garon and Jean-François Lalonde. Deep 6-DOF tracking. *IEEE transactions on visualization and computer graphics*, 2017. 1, 3

[18] Mathieu Garon, Denis Laurendeau, and Jean-François Lalonde. A framework for evaluating 6-DoF object trackers. In *ECCV*, 2018. 3

[19] Chris Harris and Carl Stennett. RAPID - a video rate object tracker. In *BMVC*, 1990. 1, 3

[20] Stefan Hinterstoißer, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary R. Bradski, Kurt Konolige, and Nassir Navab. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In *ACCV*, 2012. 12

[21] Tomáš Hodaň, Dániel Baráth, and Jiří Matas. EPOS: Estimating 6D pose of objects with symmetries. *CVPR*, 2020. 1

[22] Tomas Hodan, Pavel Haluza, Stepan Obdrzalek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-Less Objects. 2017. 2, 7, 15

[23] Tomáš Hodaň, Pavel Haluza, Štěpán Obdržálek, Jiří Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. *WACV*, 2017. 6

[24] Tomáš Hodaň, Jiří Matas, and Štěpán Obdržálek. On evaluation of 6D object pose estimation. *ECCVW*, 2016. 12

[25] Tomáš Hodaň, Frank Michel, Eric Brachmann, Wadim Kehl, Anders Glent Buch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, Caner Sahin, Fabian Manhardt, Federico Tombari, Tae-Kyun Kim, Jiří Matas, and Carsten Rother. BOP: Benchmark for 6D object pose estimation. *ECCV*, 2018. 19

[26] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders Glentbuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, and Others. Bop: Benchmark for 6D Object Pose Estimation. In *ECCV*, 2018. 2, 7

[27] Tomas Hodan, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, and Jiri Matas. BOP Challenge 2020 on 6D Object Localization. In *ECCV*, 2020. 7

[28] Tomáš Hodaň, Martin Sundermeyer, Yann Labbé, Van Nguyen Nguyen, Gu Wang, Eric Brachmann, Bertram Drost, Vincent Lepetit, Carsten Rother, and Jiří Matas. BOP challenge 2023 on detection, segmentation and pose estimation of seen and unseen rigid objects. *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024. 2, 3, 5, 6, 7, 12

[29] Yinlin Hu, Pascal Fua, Wei Wang, and Mathieu Salzmann. Single-Stage 6D Object Pose Estimation. In *CVPR*, 2020. 2

[30] Jan Issac, Manuel Wüthrich, Cristina Garcia Cifuentes, Jeannette Bohg, Sebastian Trimpe, and Stefan Schaal. Depth-based object tracking using a robust gaussian filter. In *ICRA*, 2016. 3

[31] Roman Kaskman, Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Homebreweddb: RGB-D Dataset for 6D Pose Estimation of 3D Objects. In *ICCV Workshops*, 2019. 2, 7, 16

[32] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again. In *ICCV*, 2017. 1, 2

[33] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[34] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. CosyPose: Consistent multi-view multi-object 6D pose estimation. *ECCV*, 2020. 1, 2

[35] Yann Labbé, Lucas Manuelli, Arsalan Mousavian, Stephen Tyree, Stan Birchfield, Jonathan Tremblay, Justin Carpentier, Mathieu Aubry, Dieter Fox, and Josef Sivic. MegaPose: 6D Pose Estimation of Novel Objects via Render & Compare. In *CoRL*, 2022. 1, 2, 3, 4, 5, 6, 7

[36] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An accurate O(n) solution to the PnP problem. *IJCV*, 2009. 2, 3, 4

[37] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: Deep iterative matching for 6d pose estimation. *ECCV*, 2018. 1, 2, 3

[38] Zhigang Li, Gu Wang, and Xiangyang Ji. CDPN: Coordinates-Based Disentangled Pose Network for Real-Time RGB-Based 6DoF Object Pose Estimation. In *ICCV*, 2019. 2

[39] Yunzhi Lin, Jonathan Tremblay, Stephen Tyree, Patricio A Vela, and Stan Birchfield. Keypoint-based category-level object pose tracking from an rgb sequence with uncertainty estimation. In *ICRA*, 2022. 3

[40] Xingyu Liu, Ruida Zhang, Chenyangguang Zhang, Bowen Fu, Jiwen Tang, Xiquan Liang, Jingyi Tang, Xiaotian Cheng, Yukang Zhang, Gu Wang, and Xiangyang Ji. GDRNPP. https://github.com/shanice-l/gdrnpp_bop2022, 2022. 1, 2

[41] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 1981. 1, 3

[42] Mateusz Majcher and Bogdan Kwolek. 3d model-based 6d object pose tracking on rgb images using particle filtering and heuristic optimization. In *VISIGRAPP (5: VISAPP)*, 2020. 1, 3

[43] Meta. Quest 3. https://www.meta.com/quest/quest-3/, 2023. 1

[44] Sungphill Moon, Hyeontae Son, Dongcheol Hur, and Sangwook Kim. GenFlow: Generalizable Recurrent Flow for 6D Pose Refinement of Novel Objects. In *arXiv preprint arXiv:2403.11510*, 2024. 1, 2, 3, 4, 6, 7

[45] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis: proceedings of the biennial Conference held at Dundee, year=2006, organization=Springer*. 4

[46] Van Nguyen Nguyen, Yuming Du, Yang Xiao, Michael Ramamonjisoa, and Vincent Lepetit. PIZZA: A Powerful Image-Only Zero-Shot Zero-CAD Approach to 6 DoF Tracking. 2022. 3

[47] Van Nguyen Nguyen, Thibault Groueix, Georgy Ponimatkin, Vincent Lepetit, and Tomas Hodan. CNOS: A Strong Baseline for CAD-based Novel Object Segmentation. In *ICCVW*, 2023. 4, 5

[48] Van Nguyen Nguyen, Thibault Groueix, Mathieu Salzmann, and Vincent Lepetit. Gigapose: Fast and robust novel object pose estimation via one correspondence. In *CVPR*, 2024. 1, 7

[49] Van Nguyen Nguyen, Yinlin Hu, Yang Xiao, Mathieu Salzmann, and Vincent Lepetit. Templates for 3D Object Pose Estimation Revisited: Generalization to New Objects and Robustness to Occlusions. In *CVPR*, 2022. 1

[50] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 2, 4, 5, 6

[51] Evin Pınar Örnek, Yann Labbé, Bugra Tekin, Lingni Ma, Cem Keskin, Christian Forster, and Tomas Hodan. Foundpose: Unseen object pose estimation with foundation features. In *ECCV*, 2024. 1, 2, 3, 4, 5, 6, 7, 12, 13

[52] Mustafa Özuysal, Vincent Lepetit, François Fleuret, and Pascal Fua. Feature harvesting for tracking-by-detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *ECCV*, 2006. 1, 3

[53] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-Wise Coordinate Regression of Objects for 6D Pose Estimation. In *ICCV*, 2019. 2

[54] Mahdi Rad and Vincent Lepetit. BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects Without Using Depth. In *ICCV*, 2017. 1, 2

[55] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021. 4, 6

[56] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *ICCV*, 2005. 1, 3

[57] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: an efficient alternative to SIFT or SURF. In *ICCV*, 2011. 1, 3

[58] Byung-Kuk Seo, Hanhoon Park, Jong-Il Park, Stefan Hinterstoisser, and Slobodan Ilic. Optimal local searching for fast and robust textureless 3D object tracking in highly cluttered backgrounds. *IEEE Transactions on Visualization and Computer Graphics*, 2014. 1, 3

[59] Byung-Kuk Seo and Harald Wuest. A direct method for robust model-based 3D object tracking from a monocular RGB image. In *ECCVW*, 2016. 1, 3

[60] Dave Shreiner et al. *OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1*. Pearson Education, 2009. 5

[61] Iryna Skrypnyk and David G. Lowe. Scene modelling, recognition and tracking with invariant image features. In *ISMAR*, 2004. 1, 3

[62] Manuel Stoiber, Martin Pfanne, Klaus H Strobl, Rudolph Triebel, and Alin Albu-Schäffer. Srt3d: A sparse region-based 3d object tracking approach for the real world. *International Journal of Computer Vision*, 2022. 1, 3, 6, 7, 8, 12

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

[64] Martin Sundermeyer, Tomáš Hodaň, Yann Labbe, Gu Wang, Eric Brachmann, Bertram Drost, Carsten Rother, and Jiří Matas. Bop Challenge 2022 on Detection, Segmentation and Pose Estimation of Specific Rigid Objects. In *CVPR*, 2023. 2

[65] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 2, 3, 4, 5

[66] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. Real-Time Seamless Single Shot 6D Object Pose Prediction. In *CVPR*, 2018. 1, 2

[67] Xuhui Tian, Xinran Lin, Fan Zhong, and Xueying Qin. Large-displacement 3d object tracking with hybrid non-local optimization. In *ECCV*, 2022. 1, 3, 6, 7, 8, 12

[68] Henning Tjaden, Ulrich Schwanecke, Elmar Schömer, and Daniel Cremers. A region-based gauss-newton approach to real-time monocular multiple object tracking. *TPAMI*, 2018. 2, 6, 7, 8

[69] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Stable real-time 3D tracking using online and offline information. *PAMI*, 2004. 1, 3

[70] Chen Wang, Roberto Martín-Martín, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke Zhu. 6-PACK: Category-level 6D pose tracker with anchor-based keypoints. In *ICRA*, 2020. 3

[71] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. GDR-Net: Geometry-Guided Direct Regression Network for Monocular 6D Object Pose Estimation. In *CVPR*, 2021. 1

[72] Long Wang, Shen Yan, Jianan Zhen, Yu Liu, Maojun Zhang, Guofeng Zhang, and Xiaowei Zhou. Deep active contours for real-time 6-DoF object tracking. In *ICCV*, 2023. 1, 3, 6, 7, 8, 12

[73] Philippe Weinzaepfel, Thomas Lucas, Vincent Leroy, Yohann Cabon, Vaibhav Arora, Romain Brégier, Gabriela Csurka, Leonid Antsfeld, Boris Chidlovskii, and Jérôme Revaud. Croco v2: Improved cross-view completion pre-training for stereo matching and optical flow. In *ICCV*, 2023. 4, 6

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

[75] Bowen Wen, Chaitanya Mitash, Baozhang Ren, and Kostas E Bekris. se(3)-TrackNet: Data-driven 6D pose tracking by calibrating image residuals in synthetic domains. In *IROS*, 2020. 1, 3

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

[77] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. *arXiv preprint arXiv:2312.08344*, 2023. 1, 2, 3, 4, 5

[78] Manuel Wüthrich, Peter Pastor, Mrinal Kalakrishnan, Jeannette Bohg, and Stefan Schaal. Probabilistic object tracking using a range camera. In *IROS*, 2013. 3

[79] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. *RSS*, 2018. 2, 6, 7, 8, 12, 13, 17

[80] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. 2018. 2, 7

[81] Sergey Zakharov, Ivan S. Shugurov, and Slobodan Ilic. DPOD: 6D Pose Object Detector and Refiner. In *ICCV*, 2019. 1, 2

Appendix

In this appendix, we provide details about the evaluation metrics used in the object tracking experiments (Sec. A). We also provide quantitative tracking results on the HOT3D [3] dataset (Sec. B), and additional qualitative results for both refinement and tracking (Sec. C and D).

A. Evaluation metrics for tracking

As discussed in Section 4.2 of the main paper, we use the same standard evaluation metrics as previous works [8, 62, 67, 72, 79]: Area under the Curve (AUC) for the ADD and and ADD(-S) pose error functions on YCB-V dataset, and 5*cm*-5°, and number of resets on RBOT dataset. The ADD error [20] measures the average deviation of the model vertices in the estimated pose. The ADD-S error measures the average distance to the closest model point [20, 24] as is suitable for evaluating pose estimates of symmetric objects. The AUC values are calculated by averaging the recall rates w.r.t. ADD and ADD-S for multiple thresholds ranging from 0.1 cm to 10 cm. The 5*cm*-5°score refers to the success rate of tracking poses with a translation error of less than 5 centimeters and a rotation error of less than 5 degrees. On RBOT dataset, after the initialization of the tracker with the GT pose, the tracker runs until either the recorded sequence ends or tracking was unsuccessful. In the case of unsuccessful tracking, the algorithm is re-initialized with the GT pose, and we count the total number of re-initialization.

B. Quantitative tracking results

Tab. 4 shows results on the HOT3D [3] recordings from Aria and Quest 3. Since the ground-truth annotations for the test set of the HOT3D dataset are not publicly available, we evaluate GoTrack on the training set of HOT3D. Despite not being trained on grayscale images, our method achieves a similar performance on grayscale images from HOT3D-Quest3 as on RGB images from HOT3D-Aria.

	HOT3D-Aria		HOT3D-Quest3	
	ADD	ADD-S	ADD	ADD-S
dumbbell_5lb	88.1	93.5	82.3	89.5
cellphone	84.8	90.0	81.2	85.9
mouse	88.5	93.9	86.1	91.6
keyboard	89.2	92.1	89.3	91.8
mug_white	89.0	94.6	84.5	90.9
mug_patterned	86.0	92.6	88.1	94.1
mug_patterned	87.9	93.4	88.9	94.0
bowl	62.6	90.5	64.9	88.5
plate_bamboo	62.7	80.0	63.7	77.8
flask	85.4	92.5	85.0	91.7
vase	85.5	92.4	87.8	93.4
spoon_wooden	70.4	77.3	58.4	65.5
spatula_red	65.7	72.3	57.6	66.1
potato_masher	85.4	91.4	78.8	88.0
holder_black	84.4	92.6	77.4	87.8
holder_gray	88.7	93.7	85.9	92.7
birdhouse_toy	88.4	93.8	91.2	94.4
dino_toy	87.1	92.9	85.1	92.2
dvd_remote	74.4	79.6	78.6	85.1
whiteboard_eraser	85.6	90.8	84.4	90.1
whiteboard_marker	85.2	91.5	83.8	89.1
carton_milk	89.4	93.4	85.9	92.8
carton_oj	93.6	95.9	86.2	93.1
bottle_mustard	88.4	94.1	79.5	89.4
bottle_ranch	86.8	92.8	80.9	90.3
bottle_bbq	76.8	88.5	81.2	90.9
can_soup	89.6	94.3	84.5	90.9
can_parmesan	91.0	94.6	88.9	93.8
can_tomato_sauce	90.6	94.8	85.8	92.3
food_waffles	85.1	92.2	86.3	91.6
food_vegetables	87.5	92.6	85.9	91.3
puzzle_toy	94.2	96.6	91.8	95.3
aria_small	65.7	81.1	70.9	84.9
Average	83.7	90.7	81.5	89.0

Table 4. **Tracking performance on the HOT3D dataset [3].** The accuracy is measured by AUC w.r.t. the ADD and ADD-S pose error functions for multiple thresholds ranging from 0.1 cm to 10 cm (as in Table 2 of the main paper).

C. Qualitative refinement results

Fig. 6–12 show multiple object pose refinement results (using only one iteration) on the seven core BOP datasets [28]. Each row presents one sample. The input image is shown in the first column, and the coarse pose from the fast template retrieval of FoundPose [51] is shown in the second column. The third and fourth columns show the predictions of our network (a mask of visible template pixels and an optical flow field), which can be used to remap pixels from the template to the input image as shown in the fifth column. The last column presents the initial pose (in blue), and the final pose estimated by P*n*P-RANSAC from 2D-3D correspondences which are constructed from the network predictions (in red). As can be seen in the second and third columns in all the figures, our method is robust to occlusions and accurately estimates the visible mask, which results in an accurate pose estimate. Furthermore, many of the samples (*e.g.*, the first two rows of Fig. 7) also demonstrate that our method can correct large pose deviations.

D. Qualitative tracking results

- Tracking results on the YCB-V dataset [79]: `https://youtu.be/6B1ZOQkvxH4`
- Tracking results on the HOT3D-Aria dataset [3]: `https://youtu.be/74NXe03ySdM`;
- Tracking results on the HOT3D-Quest3 dataset [3]: `https://youtu.be/bjQWS9vgPT0`.

Each video shows the inputs, the model-to-frame[1] flow, the frame-to-frame flow, the pose fitting statistics, and the

[1]The terms "model-to-frame registration" and "template-to-frame registration" are used interchangeably.

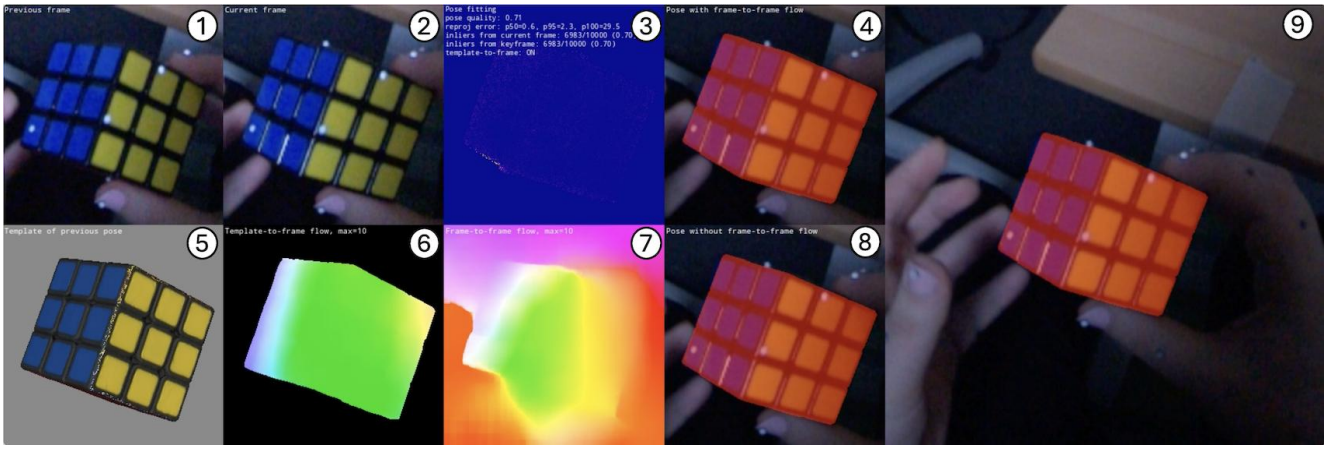

Figure 5. **Visualization of tracking.** Tiles ① and ② show crops of the previous and the current frame, respectively. Tile ⑤ shows a template rendered with the previous pose estimate. Tile ⑥ shows the masked template-to-frame flow, which is visualized for all frames but actually used only when the inlier ratio is below a threshold τ_i. Tile ⑦ shows the frame-to-frame flow which is calculated and used in every frame. Tile ③ shows a map of re-projection errors and several lines of text with information about the pose fitting. The second line of text shows the pose quality q. The third line shows the statistics of the re-projection error (50th, 75th, and 100th percentile). The fourth line shows the number of used correspondences in the current frame (A), the number of inlier correspondences in the current frame (B), and the inlier ratio in the current frame (B/A). The fifth line shows the number of inlier correspondences in the current frame (B), the number of inlier correspondences in the last keyframe (C), *i.e.*, the frame where we ran the template-to-frame registration last time, and the inlier ratio (B/C) – the template-to-frame registration is triggered if this inlier ratio is below τ_i. The last line shows whether the template-to-frame registration is triggered. Tile ④ shows a pose estimated using the full tracking approach described in Sec. 3.3 (*i.e.*, with frame-to-frame flow). Tile ⑧ shows a pose that we would obtain if we ran the template-to-frame registration at every frame without frame-to-frame flow. Tile ⑨ shows the pose estimate from ④ in the raw input frame.

estimated poses for sample sequences (see Fig. 5 for details). The poses are initialized using the template retrieval approach from FoundPose [51].

As detailed in Sec. 3.3, the model-to-frame registration is triggered when the inlier ratio is below $\tau_i = 0.8$ (*i.e.*, when less than 80% of the inlier correspondences from the last model-to-frame registration are still inliers in the current frame). The frequency of triggering the model-to-frame registration depends on dynamics of the scene – if the object pose and the viewpoint change slowly, as in YCB-V [79], the model-to-frame registration is triggered less often than in the more dynamic sequences from HOT3D [3], particularly when objects are manipulated by hands. The frequency can be reduced by lowering τ_i for the price of potentially introducing a drift from the actual pose. In every frame, we use the maximum of 10K correspondences (if more are available, we sample a random subset).

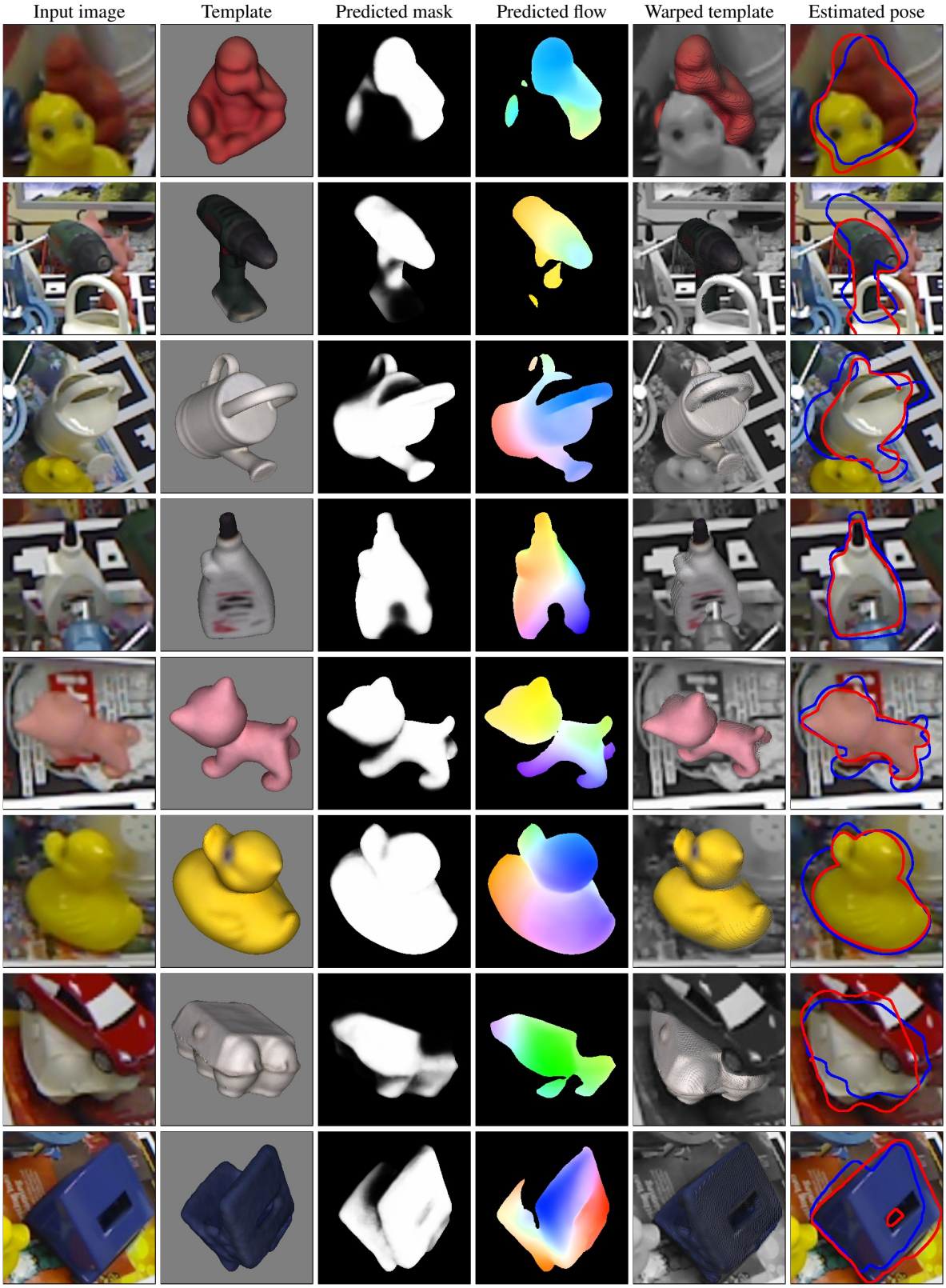

Figure 6. **Example results of the GoTrack refiner on LM-O [5].**

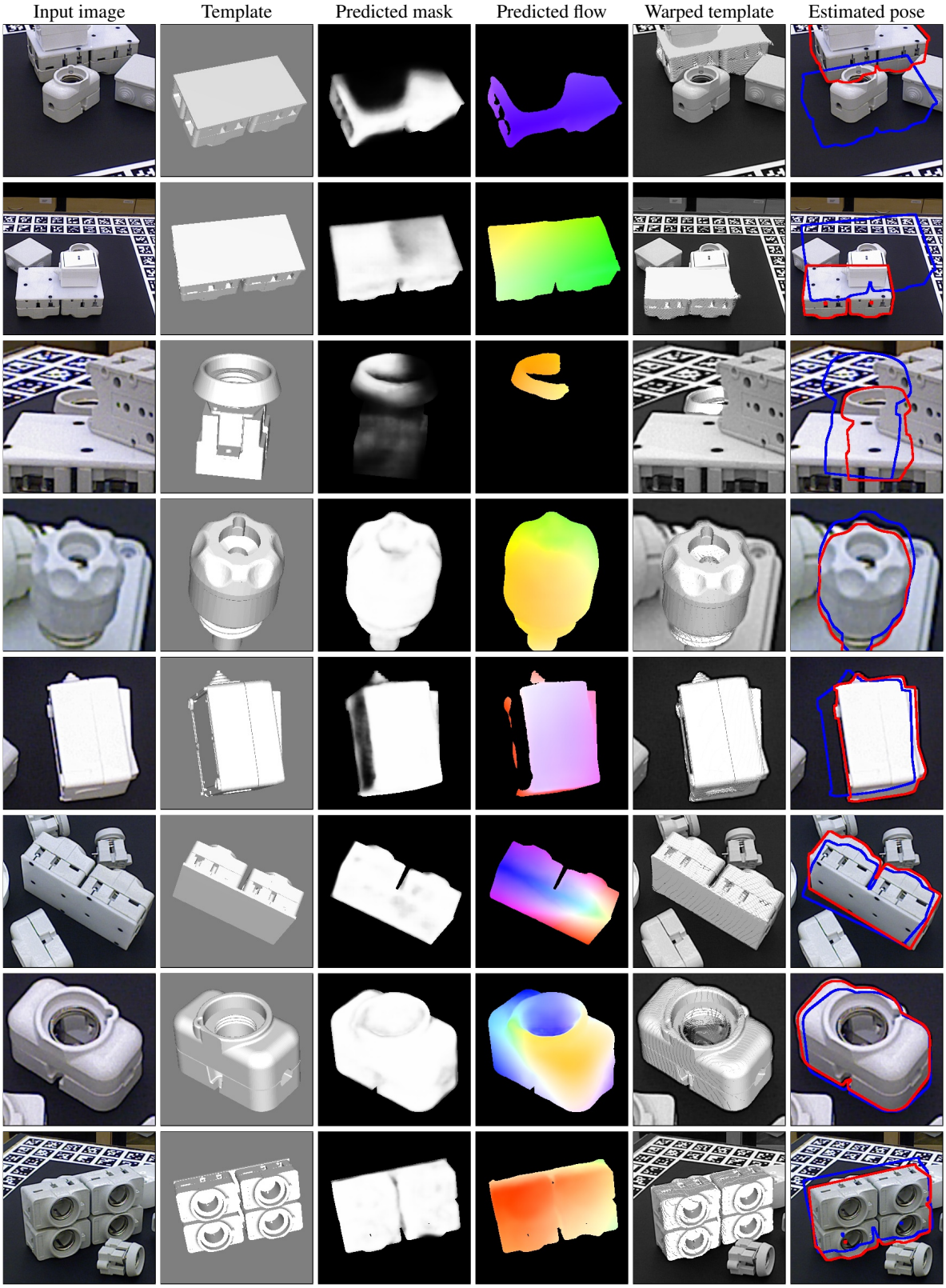

Figure 7. **Example results of the GoTrack refiner on T-LESS [22].**

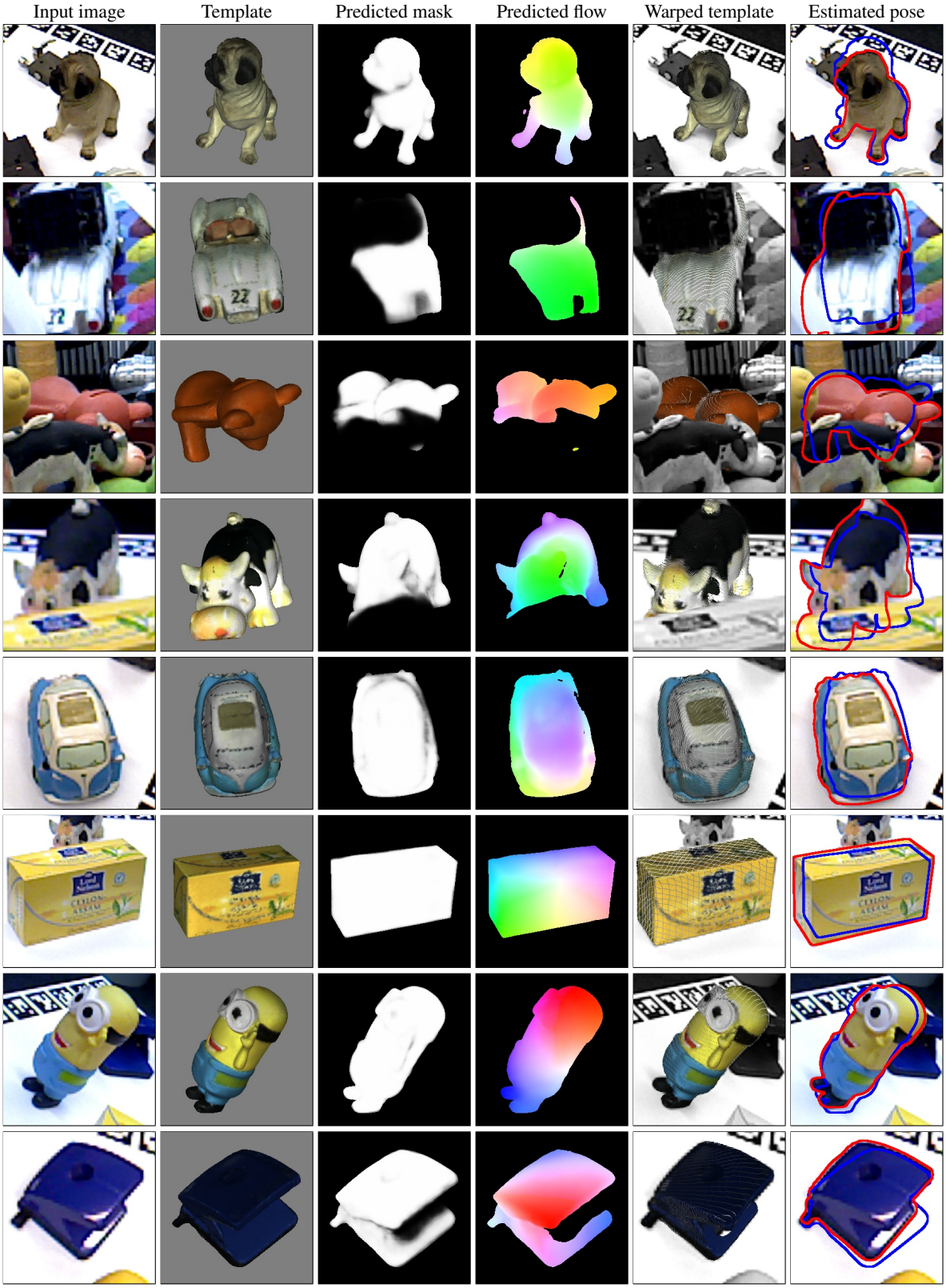

Figure 8. **Example results of the GoTrack refiner on HB [31].**

Figure 9. **Example results of the GoTrack refiner on YCBV [79].**

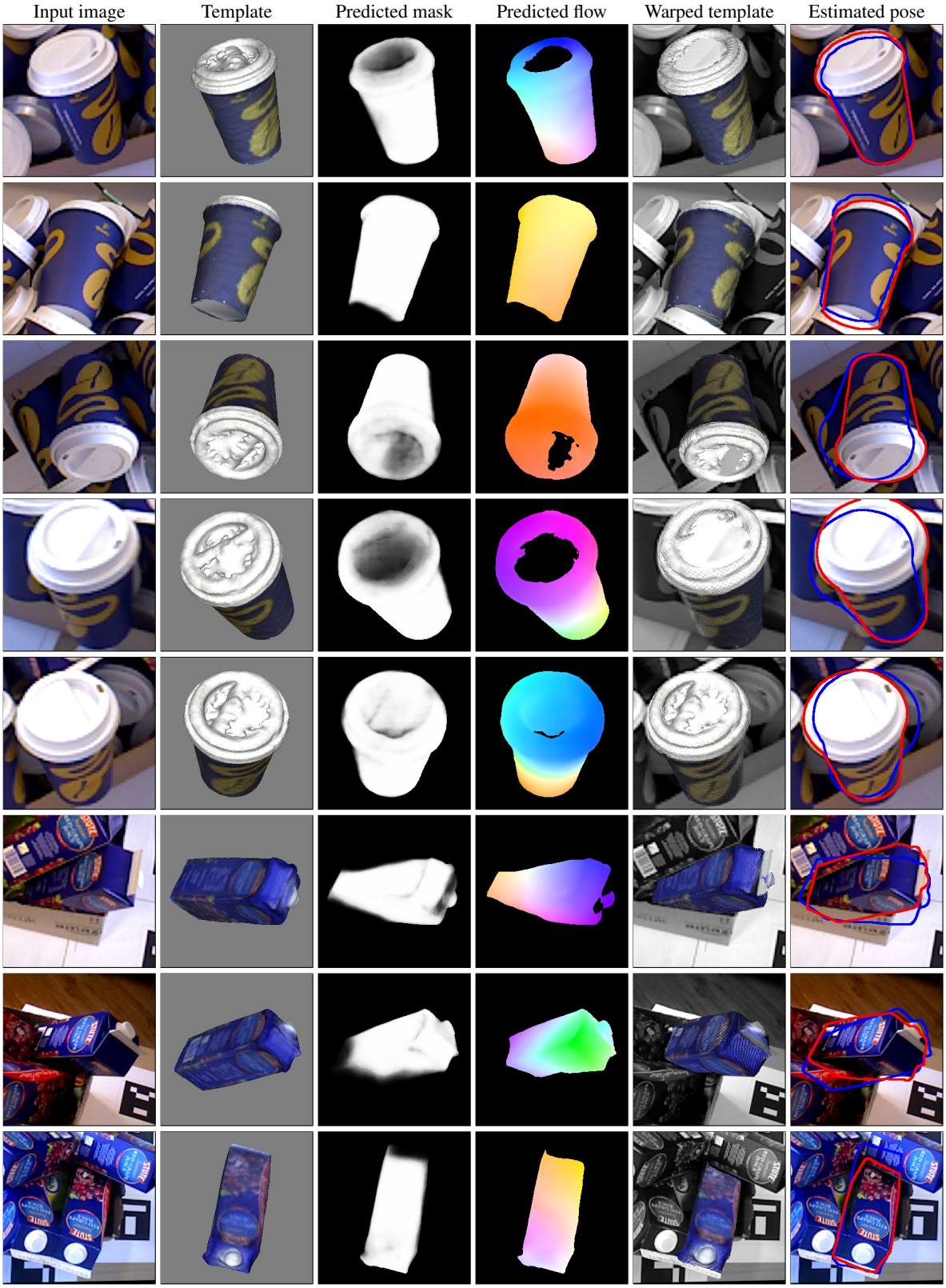

Figure 10. **Example results of the GoTrack refiner on IC-BIN [10].**

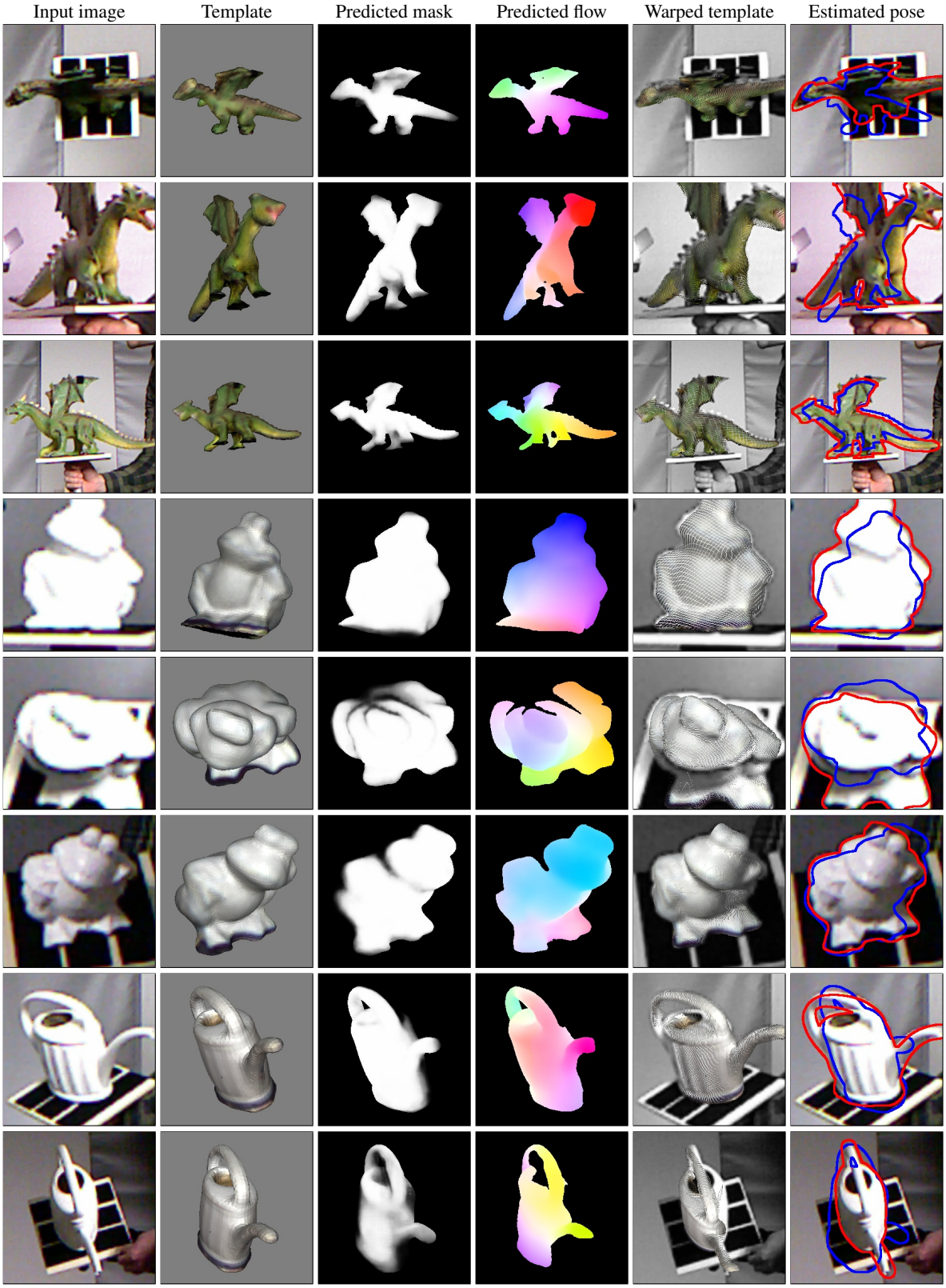

Figure 11. **Example results of the GoTrack refiner on TUDL [25].**

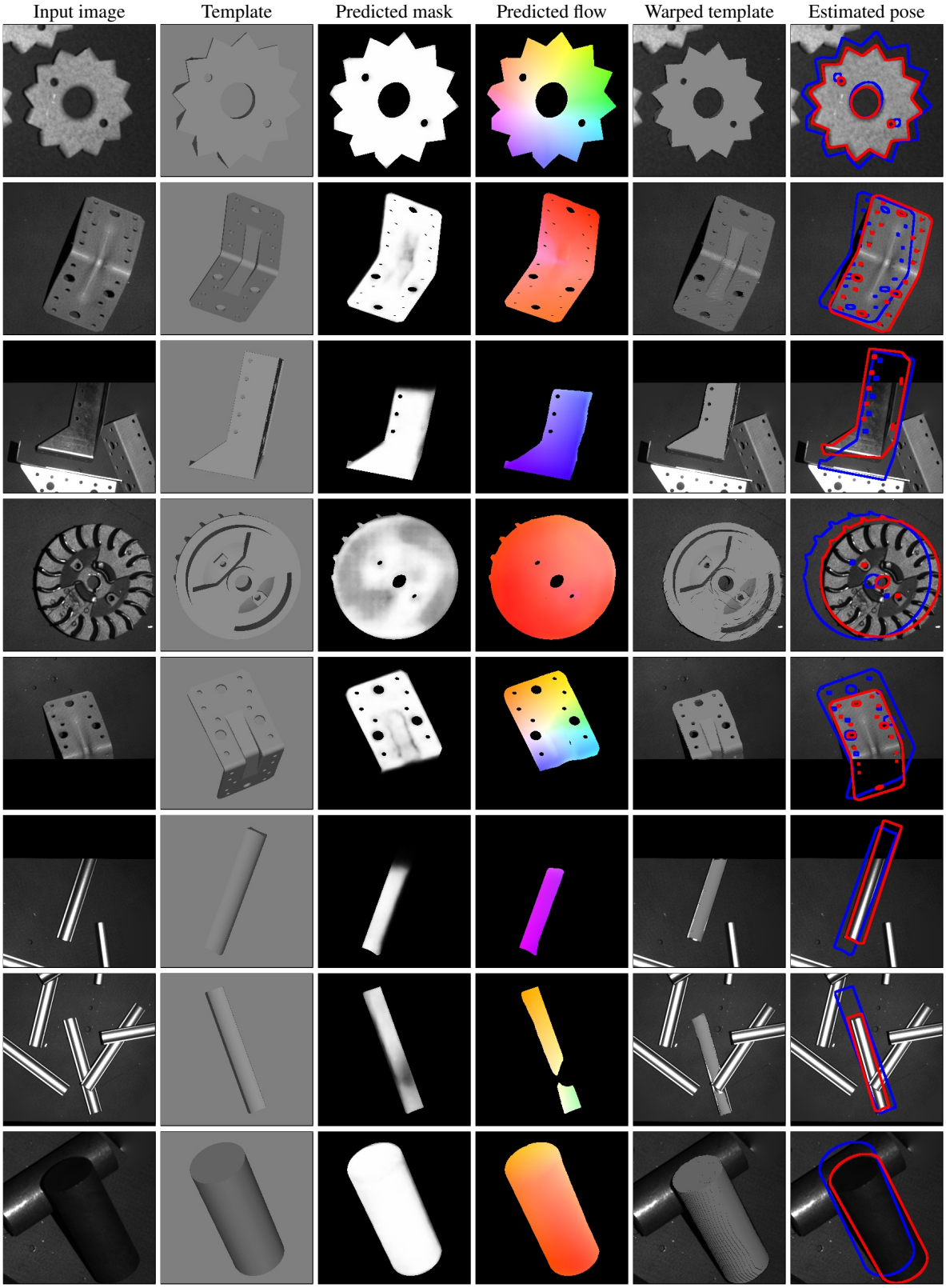

Figure 12. **Example results of the GoTrack refiner on ITODD [12].**