MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Self-Supervised Appearance-Based In-Plane Rotation Estimation

Chatterjee, Sreejani; Taylor, Nichols; Nikovski, Daniel N.

TR2026-161 October 29, 2026

Abstract

Establishing correspondences between keypoints of an object in different poses in multiple images captured by a camera is a key step in estimating its pose for the purpose of manipulation, material handling, and assembly in robotics. Full-range in-plane rotations in the image plane are particularly challenging, because the majority of keypoint detection, description, and matching methods are neither rotationally invariant nor covariant. In addition, the appearance of the object tends to undergo significant changes due to varying illumination and specular effects. We investigate the performance of four keypoint matching and two keypoint tracking methods on a representative dataset, and propose a novel memory-based method to improve matching under large in-plane rotations, where matching is done against a set of reference images whose relative poses are estimated in a self-supervised manner.

23rd International Conference on Informatics in Control, Automation and Robotics (ICINCO) 2026

© 2026 MERL. This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Research Laboratories, Inc.; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Research Laboratories, Inc. All rights reserved.

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Self-Supervised Appearance-Based In-Plane Rotation Estimation

Sreejani Chatterjee[1][a], Nichols Taylor[1,2][b], and Daniel Nikovski[1][c]
[1]*Mitsubishi Electric Research Labs, Cambridge, Massachusetts, USA*
[2]*Northeastern University, Boston, Massachusetts, USA*
schatterjee,ntaylor,nikovski@merl.com

Keywords: Robotics, Computer Vision, Pose Estimation, Machine Learning

Abstract: Establishing correspondences between keypoints of an object in different poses in multiple images captured by a camera is a key step in estimating its pose for the purpose of manipulation, material handling, and assembly in robotics. Full-range in-plane rotations in the image plane are particularly challenging, because the majority of keypoint detection, description, and matching methods are neither rotationally invariant nor covariant. In addition, the appearance of the object tends to undergo significant changes due to varying illumination and specular effects. We investigate the performance of four keypoint matching and two keypoint tracking methods on a representative dataset, and propose a novel memory-based method to improve matching under large in-plane rotations, where matching is done against a set of reference images whose relative poses are estimated in a self-supervised manner.

1 Introduction

Many tasks in robotics, such as material handling and assembly, require accurate knowledge of the pose of the manipulated objects, so that they can be grasped and manipulated. Traditional robotics systems, where a part's pose is generally ensured to be constant and repeatable by means of various jigs and fixtures, are gradually being replaced by next-generation robotics technology that allows the pose of the part to vary within the extent of the workspace, for example a worktable, a kitting tray, a bin, etc. Estimating the part's pose and adjusting the operation of the robot accordingly thus become essential technologies.

The steady advance of imaging technology now allows image frame capture at rates approaching that of the desired control rates for industrial robots and at resolutions sufficient for accurate pose estimation. Consequently, a large number of methods for pose estimation from camera input have been proposed (Liu et al., 2026), making various assumptions about what kind of additional information about the target object is available. Such information can be of major benefit to estimation methods. For example, if CAD models are available, a rendered view of the object can be matched against the current image. However, expecting CAD models is not practical or reasonable for most deployments. Similarly, if images of the object are labeled with its true pose in those images, a supervised learning method can be used to predict the pose of the object in a new image. However, producing such a labeled dataset is laborious and not economically appealing.

A much more practical alternative are self-supervised methods, where a dataset of images of the object moving in the scene is collected, but without pose labels. If the sequence of images can be analyzed to understand how the object looks in various poses, and furthermore the pose of the object in these images can be estimated based on its temporal evolution in successive images, most or all of human effort in labeling images or constructing models can be eliminated, significantly reducing the time, cost, and effort in deploying robotic systems that rely on pose estimation. Due to its economic significance, this is the primary problem setting we are considering.

Formally, given a sequence of T images I_k, $k = 1,\ldots,T$ from a stationary camera that contain a target object, and a test image I from the same camera, we want to estimate the relative pose of the object with respect to a reference image I_0. This reference image can be one of the images in the sequence, and w.l.o.g. we will assume that $I_0 = I_1$, that is, relative pose is measured with respect to it. (In this problem setting, if there are no labels of the pose in the images I_k, we

[a] https://orcid.org/0009-0001-0949-4233
[b] https://orcid.org/0009-0002-5669-450X
[c] https://orcid.org/0000-0003-2919-645X

cannot estimate the absolute pose of the object, but only its relative pose with respect to some reference pose, subject to symmetries; for a symmetric object, returning a correct estimate corresponding to any of the symmetric views will be deemed correct.)

Two main variants of this problem exist. In the simpler variant, the motion of the object is planar, so its pose belongs to $SE(2)$ and has three degrees of freedom (DoF). The second variant is when the object moves in space with full 6 DoF, so its pose belongs to $SE(3)$. Similarly, the images I_k can be 2D, collected from an RGB camera, or also have a depth map, captured by an RGB-D camera.

When the motion of the object is planar and an RGB camera is used to estimate its pose, the most advantageous position of the camera is overhead and fronto-parallel, with its optical axis perpendicular to the plane of motion (or close to it). In that case, assuming that the size of the object is much smaller than its distance to the camera (weak perspective), the object's points projected onto the image plane undergo an affine transformation when the object varies its pose in the plane. If the affine transformation can be estimated from the images of the object in the two poses, its rotation and translation in the plane of motion can be recovered.

It is thus of primary interest to be able to establish correspondences between the same points of the object in two different images taken by an overhead fronto-parallel camera. Our objective in this study is to find reliable ways to perform this. The main challenge in this setting is handling full-range in-plane rotations. Section 2 reviews several methods for matching images of rigid objects, both keypoint-based and template-based. The majority of these methods do nor perform well for full-range in-plane rotations. Many keypoint-matching methods perform poorly, because their keypoint detectors and descriptors are neither rotationally invariant nor covariant. One possible explanation for this is that they are primarily used for applications such as stereo vision and image retrieval, where large rotations are not typical. Similarly, template-based methods that attempt to register the two images by cross-correlation do not handle rotations easily. The performance of several keypoint-based methods on a representative data set of objects from the YCB set is investigated experimentally in Sections 3 and 4.

To address the general inability of keypoint matching methods to establish correspondences over large rotations, we propose a memory-based method for relative pose estimation where matching is performed not with respect to a single reference image, but to a set of such images. Section 5 demonstrates high accuracy of matching and relative pose estimation on the same YCB-derived data set. However, when matching to a set of reference images, the relative pose of the object in each of them needs to be known. We propose to estimate it using the same keypoint-matching methods across successive image frames in the reference set. Section 5 presents results from using this approach on the same data set. Section 6 concludes the paper is describes how this research will be continued.

2 Related Work

Several approaches to solving the relative pose estimation problem exist in the setting where neither CAD models of the target object nor pose labels are available (Liu et al., 2026). The general principle of operation of such methods is to compare the appearance of the object in an unknown pose to that of the object in a reference pose. The difference among the methods is in which elements of the appearance are compared. The dominant approach is based on establishing correspondence between distinctive elements (keypoints) in the two images. Another, template-matching approach, is to match the entire images (bitmaps of masked objects), possibly after mapping them to a derived feature space.

Another major difference among existing methods is determined by whether the relative pose of the object in the two images is assumed to be small or large. The former case applies when the two images belong to a sequence capturing the smooth motion of the object in time. In this case, various tracking methods can be used that leverage the assumption of close-range motion to effectively find it. The latter applies when the object in the two images is in general position and the relative pose is arbitrary large. For this case, a tracking method cannot be used.

Classical matching methods based on keypoint matching include the SIFT method (Lowe, 2004) that detects distinctive patches in the scale-space pyramids of the two images, computes scale and rotation invariant descriptors of the patches, and finds nearest neighbors between patches from the two images in descriptor feature space. Faster variants, such as SURF (Bay et al., 2006) and ORB (Rublee et al., 2011) exist.

Modern matching methods based on deep neural networks (DNN) attempt to improve the three stages of the matching approach: keypoint detection, keypoint description, and keypoint matching. SuperPoint (DeTone et al., 2018) learns descriptors that are suitable for matching in the form of feature maps computed by a DNN. XFeat (Potje et al., 2024) is a sim-

ilar method that aims to compute the descriptor (features) quickly. R2D2 (Revaud et al., 2019) learns features that are not only repeatable (likely to be found in both images) but also reliable (not likely to be confused with other, similarly looking patches). These descriptors can be matched by dedicated matching DNNs optimized for the purpose, including SuperGlue (Sarlin et al., 2020) and LightGlue (Lindenberger et al., 2023). DISK is an entire differentiable pipeline for detection, description, and matching that optimizes the accuracy of matching by what is essentially policy-gradient reinforcement learning (Tyszkiewicz et al., 2020).

On the tracking/optical flow slide, the well-known classical Kanade-Lucas-Tomasi (KLT) tracker is based on detecting distinctive keypoints and estimating their short-range motion (Lucas and Kanade, 1981). It detects Shi-Tomasi distinctive keypoints of the object, such as corners. Modern DNN-based methods include TAPIR (Doersch et al., 2023) and CoTracker3 (Karaev et al., 2024). An advantage of the latter is that it learns how sets of keypoints move jointly and can potentially track points through temporary occlusions.

Correspondence-based methods are effective when the object has sufficient texture to identify distinctive visual elements (corners, blobs, etc.) On the other hand, template-matching methods can be effective when no such texture is available. The simplest template-based method is to compute the cross-correlation between the bitmaps and detect its peak, effectively registering the two images. This can be done efficiently in the frequency domain by means of the Fourier-Mellin method (Reddy and Chatterji, 1996). If the shapes of the object bitmasks have orientation, computing their principal axes is a way to obtain an initial estimate of their relative rotation.

DNN-based methods have been used for template matching, too. Many such methods assume access to the ground truth about the registration transformation, which is not the setting we are interested in. Of more relevance are unsupervised learning methods, and in particular methods that predict a single affine transformation, as opposed to pixel-level alignment, which is more applicable to deformable objects. Affine registration networks learn to predict the correct affine transformation that aligns two images fed as its inputs (Jaderberg et al., 2015). Some networks use Siamese encoders to transform the raw pixels to a feature space where matching and registration is easier (De Vos et al., 2019).

It should also be noted that for a rotationally symmetric object, no method can identify the correct orientation of the object, because it is ambiguous. However, for the purpose of object manipulation, such symmetry makes the determination of orientation generally unnecessary, so any reported orientation, or a chosen canonical one, can be considered a correct and useful estimate.

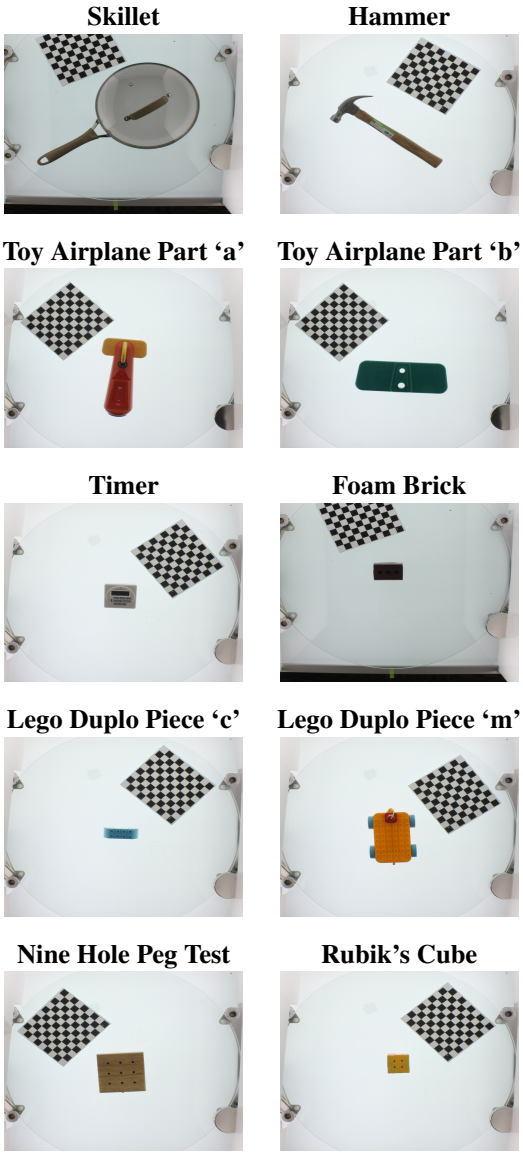

Figure 1: Set of YCB objects selected for evaluation.

3 Experimental Evaluation

A keypoint matcher or tracker is the front end for pose estimation and visual servoing. Both estimate the object’s motion from where its keypoints land in the current image relative to a reference, so the front end has

to return correct correspondences as the object moves. In-plane rotation is the case that breaks this. Most descriptors lose repeatability as the object turns, and a tracker can drift or lock onto the wrong point once the appearance rotates far from the reference. A method that holds correspondence through a large rotation is therefore the one worth putting in front of a pose estimator or a controller.

To find that method, we need rotation ground truth: an object turned by a known angle so the recovered rotation can be compared to the true one. The YCB Object and Model Set (Calli et al., 2017) provides this. Each object was captured on a motorized turntable under a fixed overhead camera (the N5 view), with one frame every 3° over a full rotation, yielding 120 frames per object. The turntable frame index determines the rotation of each frame, so the in-plane angle is known to the resolution of the table and requires no pose estimation. We selected 10 objects for evaluation, shown in Figure 1, out of a total of 125, aiming to cover a wide range of surface textures and symmetries for evaluation.

Pre-processing separates the object from its surroundings. The dataset provides a segmentation mask for each frame. (In a real deployment, if we use an overhead stationary camera, object masks can be obtained relatively easily by means of background subtraction.) We keep the object pixels and paint the background a single flat gray, so every keypoint found by a detector lands on the object and no static background feature can enter a correspondence. Frames are undistorted with the N5 camera intrinsics and kept at native resolution.

3.1 Methods and Protocol

We evaluated six methods, four per-frame matchers and two trackers. The matchers are SIFT (Lowe, 2004) with the ratio test, and three learned descriptors, XFeat (Potje et al., 2024), R2D2 (Revaud et al., 2019), and DISK (Tyszkiewicz et al., 2020). The discovered correspondences were used to compute a partial-affine transform, from which the rotation angle was recovered. RANSAC was used for robust estimation in OpenCV (Bradski and Kaehler, 2008).

The three learned descriptors are not rotation invariant, so their match quality falls quickly as the object turns away from the reference. To extend them to large rotations we use a rotated-reference scheme, denoted by +RotRef. To this end, we generate $M = 12$ rotated copies of the reference frame I_0 at 30° spacing, and match each incoming frame against all M of them, keeping the copy that yields the most RANSAC inliers. Detections from the chosen copy are rotated back into the coordinate frame of I_0 and tied to the nearest original keypoints, so keypoint identity is preserved across the turn. This recovers correspondence under large rotation at the cost of M matching operations per frame. SIFT is already rotation invariant and does not need rotated references. The two trackers tested are the KLT optical-flow tracker (Lucas and Kanade, 1981; Suhr, 2009) and the CoTracker3 temporal tracker (Karaev et al., 2024).

For every method, the first frame is the reference and each later frame is matched or tracked back to it. The in-plane rotation per frame is computed from a partial-affine fit (similarity transform) under RANSAC, refit on the inliers, and the per-frame error is the wrapped difference to the turntable angle. Turntable handedness is not assumed, so the global sign is chosen per object and method to agree with ground truth. We also rank keypoints by a reliability score, defined as the product of how many frames a keypoint survives and how often it is a RANSAC inlier, and repeat every measurement on the full keypoint set and on the top 10% of keypoints.

3.2 Isolating Rotation From Capture

A turntable frame mixes the effect of rotation with everything else that impacts the object image as it moves, including changes in illumination, shifts in focus, reflections moving across the surface, and parts of the object rotating out of view. When a method fails on such a sequence, the failure could come from the rotation or from any of these other factors. To isolate these effects, we synthetically generated a second sequence for each object by rotating the reference frame I_0 in the same 3° steps. This synthetic sequence carries identical rotation ground truth but no frame-capture effects. Running the same six methods on it measures how each one responds to pure in-plane rotation. The gap between the real and the synthetic result then shows what the camera effects, rather than the rotation, cost each method.

4 Experimental Results

For each method, we compute the absolute rotation error of every frame aggregated over the ten objects. Figure 2 compares the real and the synthetic distribution of each method, and Table 1 lists the median and the fraction of estimates with more than 90° error for each.

It is immediately obvious that the unmodified XFeat, R2D2, and DISK matchers peform very poorly at this task, most likely because they are not rotation-

ally invariant and any significant rotation angle foils them. SIFT, in contrast, performs very well. On the synthetic sequences, SIFT and the other three matchers modified to handle large rotations recover rotation almost perfectly. Every median stays below one degree, from 0.00° for SIFT to 0.47° for DISK+RotRef. This demonstrates that they can, in principle, handle full in-plane rotation very well.

On the real sequences, where frame capture effects are present, the same four matchers still achieve low overall medians, between 1.1° and 2.4°. However, between 4% and 21% of their real frames have errors larger than 90° (the points climbing toward 180° in Figure 2). These failures are not present in the corresponding synthetic sequences, where the errors return to the synthetic floor. This suggests that rotation itself is not the main cause for failure. Instead, the failures come from real capture effects and object appearance. The likely cause is surface properties that a real turn exposes and a synthetic rotation does not, shiny or partly transparent materials whose bright and see-through regions move as the object turns, and repetitive or near-symmetric texture where one part of the surface resembles another and the match can land on the wrong but visually similar part of the object. We do not measure these factors per object here. The conclusion suggested by the data is that the extra error is caused by capture effects and not by rotation. When a method fails, it typically tracks for part of a turn and then jumps by 180° onto a face or region that looks similar, so the failed frames pile up near 180° rather than scattering, and the 95th percentile reaches 179°.

Surprisingly, CoTracker3 is the one method that fails in both cases. Its median error is high on both synthetic and real data, near 26° and 56°, and its whole distribution stays high instead of dropping. Pure rotation already defeats it, so the cause here is the tracker, not the capture. KLT follows rotation well but drifts over the long sequence, which lifts its real median above that of the matchers.

Keeping only the top 10% of keypoints helps the matchers a little and leaves CoTracker3 where it was. The matcher medians barely change, while CoTracker3 rises to 44° on synthetic and 53° on real, since dropping keypoints cannot fix a track that is sliding as a whole.

Overall, SIFT has the lowest error, followed by XFeat+RotRef. Among the trackers, KLT has much better accuracy than CoTracker3. The median error is robust to a few bad frames, so a method that tracks well for most of a turn keeps a low median even if some frames are off. This figure reports per-frame accuracy in aggregate, not the angle at which a method first loses the track.

Table 1: Rotation error over all per-frame estimates for the 10 YCB objects, top 10% of keypoints, in degrees. "Median" is the median absolute error over every estimate. "$\geq 90°$" is the share of estimates whose error exceeds 90°, the rate at which a method lands on the wrong rotation branch. The synthetic columns hold the object fixed and rotate the reference frame, the real columns use the turntable images. Each learned descriptor appears twice, once matched against a single reference and once with the rotated-reference bank. Both members of a pair start from the same frame-0 detections and use the same matcher, so the bank is the only difference between them. Estimates on which a method returns no fit are excluded, so coverage varies by method, from 851 of 1190 frames for R2D2 to 1190 for XFeat+RotRef and KLT. Lower is better.

Method	Synthetic		Real	
	Median	$\geq 90°$	**Median**	$\geq 90°$
SIFT	0.01	0%	0.66	4%
XFeat	77.77	41%	88.39	43%
R2D2	85.26	45%	84.95	46%
DISK	89.11	48%	88.30	48%
XFeat+RotRef	0.13	0%	0.89	14%
R2D2+RotRef	0.25	2%	1.25	18%
DISK+RotRef	0.45	5%	1.47	21%
KLT	0.51	0%	1.06	0%
CoTracker3	21.23	23%	60.39	37%

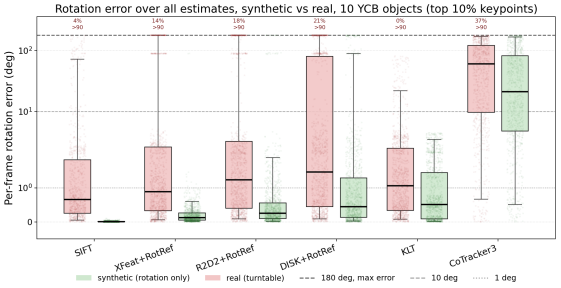

Figure 2: Rotation error over all per-frame estimates, aggregated over the 10 YCB objects, for the top 10% of keypoints. The green box is the synthetic sequence, built by rotating the reference frame, the pink box is the real turntable sequence. Each box shows the interquartile range, the whiskers reach the 5th and 95th percentiles, and the faint points are the individual estimates. The dashed lines mark 180° and 10°, the dotted line 1°. The percentage above each method is the share of its real estimates past 90°.

5 Matching to a Memory Set

A single reference frame cannot stand in for the object across a full turn, which is why matching to it fails. For this reason, we propose to keep many reference frames instead, each with a known rotation angle, and match every new frame against the whole set. We call this set the memory set (MS). A query frame is matched against every labeled frame in the MS, the matches are ranked by their inlier count, and the angle

of the strongest match is returned. This is a form of memory-based learning (MBL). It needs no model of the object and no training, only a labeled set of past views and a matcher.

To evaluate whether this idea is beneficial, on the same dataset of ten object sequences, we split each object's sequence into two interleaved halves, essentially producing a training and testing sets for MBL. In this way, a query (testing) image is never matched against itself. Both halves cover the full turn at effective 6° increments. Every error reported below is the recovered angle measured against the true turntable angle, pooled over the same ten objects, using the single best match. A textured object at full resolution produces tens of thousands of keypoints, and the match cost grows with the square of that count, so all four matchers in this section run at a fixed working resolution of 1600 pixels on the long side.

5.1 The MS with True Labels

Figure 3 (green) and Table 2 report the accuracy when the reference angles are known, i.e., the true turntable angles. Every matcher stays below half a degree at the median, from 0.11° for SIFT to 0.41° for XFeat. None of them have catastrophic failures – the fraction of frames with errors exceeding 90° is zero for all four, down from the 4 to 21 percent the same matchers showed against a single reference in Table 1. The reason is likely that for any query pose, some MS frame is close enough in appearance to match it cleanly, so catastrophic failures that were often present in the single-reference-frame case no longer appear.

This result depends on knowing the rotation angle of every MS frame. The turntable gives these angles. Without it, they are unknown, and the next section addresses our proposal for how to deal with that case.

5.2 Labeling the MS Frames Without a Turntable

Our proposal is to estimate labels for the frames in the MS in a self-supervised learning manner and thus remove the dependence on the turntable. The frames in the MS arrive as a sequence, so the rotation between successive frames is relatively small. We can estimate that small rotation with the same matcher or tracker used at run time, and chain the estimates from the first frame onward, thus computing the rotation angle for each MS frame incrementally. Frame one is the reference at zero degrees of rotation. Each later frame takes the angle of the frame before it and adds the incremental rotation between them. We propose two guardrails to keep the chain from failing catastrophically due to a single incorrect estimate of the incremental rotation. First, the per-step rotation is a robust median over the few most recent frames rather than a single estimate from the previous frame, and a small-jump gate rejects any step too large to be a real motion between adjacent frames, which is the signature of a flip of orientation. No single step is catastrophic once both are in place.

However, the incremental estimate is still prone to drifts. Each step carries a small error in a consistent direction, and these errors accumulate along the turn. Figure 3 (pink) is the same MS scored with these self-built labels in place of the turntable angles. The medians rise above the green boxes, SIFT to 0.65°, XFeat to 5.26°, R2D2 to 9.00°, and DISK to 19.95°. The rise follows how strongly each matcher's per-step estimate is biased, not how often it fails outright. SIFT's per-step bias is near zero, so its incremental estimate barely deviates from the true one, while DISK's accumulates into a twenty-degree offset over a full turn.

The comparison between the two label sources is the result of interest. Self-built labels cost some accuracy – the gap between the pink and green boxes – but the MS still avoids catastrophic failures. SIFT and XFeat hold the share of frames with errors past 90° at zero under both label sources. R2D2 and DISK let it rise only to 5 and 9 percent, far below the 18 and 21 percent they showed against a single reference. Moreover, arguably, a drifted but flip-free estimate is the right trade-off for visual servoing, where consistent errors in the estimates of configuration variables generally cancel when the visual feedback error is computed. That is, the controller regulates around a consistent estimate, so a constant offset in the labels only shifts the target. A flip can destabilize the control loop, and a constant offset generally will not.

5.3 Accuracy and Cost

Accuracy and run-time cost rank the matchers differently. Figure 4 splits the per-frame cost into the time to encode a frame (detect keypoints and compute their descriptors) and the time to match it against the MS. SIFT matches cheaply, about 6 ms against the whole MS, but its keypoint detection is the slowest of the four at about 110 ms a frame. XFeat encodes in about 12 ms and matches in about 40 ms. R2D2 and DISK both encode near 100 ms and match at 29 and 69 ms. Adding the encode and match times, XFeat is the fastest end-to-end at about 52 ms per frame, SIFT next at about 116 ms, with R2D2 and DISK above both.

Thus, the accuracy winner and the speed winner are different methods. SIFT is the most accurate by a clear margin, both as a labeler and against the

Table 2: MS rotation error over all per-frame estimates for the 10 YCB objects, best match, in degrees. The turntable columns label the MS frames with the true angles, the incremental columns with the self-estimated labels. "Median" is the median absolute error over every estimate. "$\geq 90°$" is the share past 90°, the flip rate. Lower is better.

	Turntable		Incremental	
Method	**Median**	$\geq 90°$	**Median**	$\geq 90°$
SIFT	0.11	0%	0.65	0%
XFeat	0.41	0%	5.26	0%
R2D2	0.32	0%	9.00	5%
DISK	0.27	0%	19.95	9%

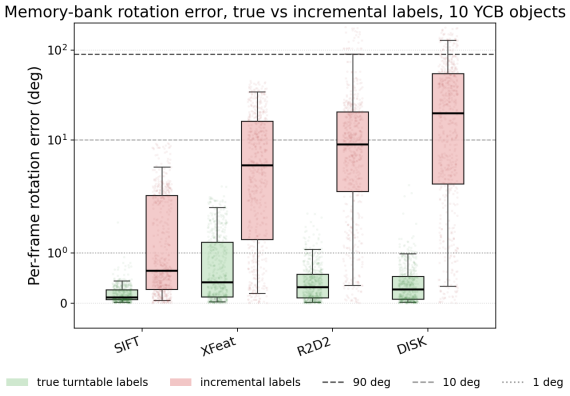

Figure 3: MS rotation error over all per-frame estimates, aggregated over the 10 YCB objects, best match only. The green box uses the true turntable angles to label the MS frames, the pink box the self-supervised incremental labels. Each box shows the interquartile range, the whiskers reach the 5th and 95th percentiles, and the faint points are the individual estimates. The dashed lines mark 90° and 10°, the dotted line 1°. The gap between the two boxes for a method is the accuracy cost of dropping the turntable.

MS. XFeat is the fastest end-to-end and the second most accurate. The choice between them is the real trade-off for a deployment, and it depends on whether the computational budget is set by the labeling pass, which runs once, or by the per-frame cost at run time.

6 Conclusion and Future Work

We have investigated the performance of several methods for keypoint matching and tracking for the purpose of estimation of full-range in-plane object rotations, as would be required in many robotic applications. On a representative set of 10 YCB objects, we discovered that most existing methods, both classical such as SIFT, as well as modern DNN-based ones such as XFeat, R2D2, and DISK, often fail to identify correct correspondences. One possible explanation for this is that these methods were originally designed

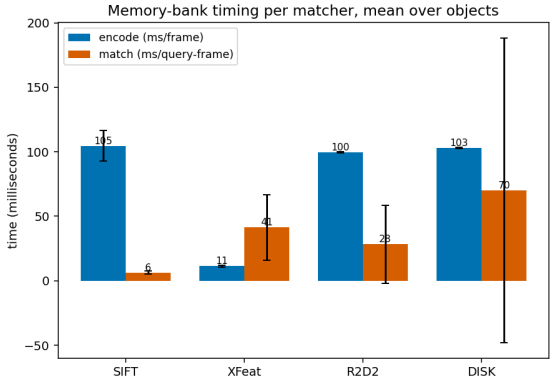

Figure 4: Per-frame timing of the MS, mean over the 10 YCB objects, at a working resolution of 1600 pixels. The blue bar is the time to encode one frame, the orange bar the time to match one query frame against the whole MS. End-to-end cost is the two added. SIFT is cheapest to match but slowest to encode, XFeat the reverse and fastest end to end. The wide spread on the DISK match bar comes from one textured object that detects far more keypoints than the rest, and the match cost grows with the square of the keypoint count.

for other computer vision applications, such as stereo vision, where large in-plane rotations are not typical or needed. Augmenting the reference pose by rotating it at multiple 30^o increments partially resolved the absence of rotational invariance in most keypoint descriptors, but still resulted in multiple catastrophic large errors in pose estimation (between 4% and 21% on the experimental dataset). This demonstrated that a single reference image cannot stand in for the object across a full turn, so this kind of data augmentation is not completely effective. Experiments on a synthetically-generated dataset for the same objects suggested that changes in appearance and illumination might be the cause.

To address this, we proposed a method for memory-based learning of the rotation angle that uses a sequence of reference images captured under continuous motion of the object. Because this method needs correct estimates of the object's pose in each memory frame, which will generally not be available during deployment, we propose to use the same relative pose estimation method over successive image frames to incrementally build the rotational angle estimates in a self-supervised learning manner. At least two of the feature matching methods (SIFT and XFeat) estimated the rotation angle based on these labels without any catastrophic failures and are thus good candidates for industrial deployment. They also happen to present different trade-offs between speed and accuracy of estimation.

So far, this study has evaluated keypoint-based

matching and tracking methods. Next, we plan to evaluate also template-based methods, applying the same idea about an entire memory set of reference images whose rotation angle is estimated in an incremental self-supervised manner. The true test of the usefulness of the proposed method will come from its use in typical robotic tasks, such as pick-and-place manipulation tasks, as well as visual servocontrol. Another, more challenging task to investigate is whether pose estimation can be performed while the object is in the robot's gripper, possibly occluded partially.

REFERENCES

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer.

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.".

Calli, B., Singh, A., Bruce, J., Walsman, A., Konolige, K., Srinivasa, S., Abbeel, P., and Dollar, A. M. (2017). Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268.

De Vos, B. D., Berendsen, F. F., Viergever, M. A., Sokooti, H., Staring, M., and Išgum, I. (2019). A deep learning framework for unsupervised affine and deformable image registration. *Medical image analysis*, 52:128–143.

DeTone, D., Malisiewicz, T., and Rabinovich, A. (2018). Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236.

Doersch, C., Yang, Y., Vecerik, M., Gokay, D., Gupta, A., Aytar, Y., Carreira, J., and Zisserman, A. (2023). Tapir: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10061–10072.

Jaderberg, M., Simonyan, K., and Zisserman, A. (2015). Spatial transformer networks. *Advances in neural information processing systems*, 28.

Karaev, N., Makarov, I., Wang, J., Neverova, N., Vedaldi, A., and Rupprecht, C. (2024). CoTracker3: Simpler and better point tracking by pseudo-labelling real videos. *arXiv preprint arXiv:2410.11831*.

Lindenberger, P., Sarlin, P.-E., and Pollefeys, M. (2023). Lightglue: Local feature matching at light speed. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 17627–17638.

Liu, J., Sun, W., Yang, H., Zeng, Z., Liu, C., Zheng, J., Liu, X., Rahmani, H., Sebe, N., and Mian, A. (2026). Deep Learning-Based Object Pose Estimation: A Comprehensive Survey: J. Liu et al. *International Journal of Computer Vision*, 134(2):81.

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

Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679.

Potje, G., Cadar, F., Araujo, A., Martins, R., and Nascimento, E. R. (2024). Xfeat: Accelerated features for lightweight image matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2682–2691.

Reddy, B. S. and Chatterji, B. N. (1996). An FFT-based technique for translation, rotation, and scale-invariant image registration. *IEEE transactions on image processing*, 5(8):1266–1271.

Revaud, J., De Souza, C., Humenberger, M., and Weinzaepfel, P. (2019). R2d2: Reliable and repeatable detector and descriptor. *Advances in neural information processing systems*, 32.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision*, pages 2564–2571. Ieee.

Sarlin, P.-E., DeTone, D., Malisiewicz, T., and Rabinovich, A. (2020). Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947.

Suhr, J. K. (2009). Kanade-lucas-tomasi (klt) feature tracker. *Computer Vision (EEE6503)*, pages 9–18.

Tyszkiewicz, M., Fua, P., and Trulls, E. (2020). Disk: Learning local features with policy gradient. *Advances in neural information processing systems*, 33:14254–14265.