SLAM-Supported Self-Training for 6D Object Pose Estimation

Ziqi Lu¹, Yihao Zhang¹, Kevin Doherty¹, Odin Severinsen¹, Ethan Yang¹, John Leonard¹

Abstract—Recent progress in object pose prediction provides a promising path for robots to build object-level scene representations during navigation. However, as we deploy a robot in novel environments, the out-of-distribution data can degrade the prediction performance. To mitigate the domain gap, we can potentially perform self-training in the target domain, using predictions on robot-captured images as pseudo labels to fine-tune the object pose estimator. Unfortunately, the pose predictions are typically outlier-corrupted, and it is hard to quantify their uncertainties, which can result in low-quality pseudo-labeled data. To address the problem, we propose a SLAM-supported self-training method, leveraging robot understanding of the 3D scene geometry to enhance the object pose inference performance. Combining the pose predictions with robot odometry, we formulate and solve pose graph optimization to refine the object pose estimates and make pseudo labels more consistent across frames. We incorporate the pose prediction covariances as variables into the optimization to automatically model their uncertainties. This automatic covariance tuning (ACT) process can fit 6D pose prediction noise at the component level, leading to higher-quality pseudo training data. We test our method with the deep object pose estimator (DOPE) on the YCB video dataset and in real robot experiments. It achieves respectively 34.3% and 17.8% accuracy enhancements in pose prediction on the two tests. Our code is available at https://github.com/520xyxyzq/slam-super-6d.

I. INTRODUCTION

State-of-the-art object 6D pose estimators can capture object-level geometric and semantic information in challenging scenes [1]. During robot navigation, an object-based simultaneous localization and mapping (object SLAM) system can use object pose predictions, with robot odometry, to build a consistent object-level map (e.g. [2]). However, as we deploy the robot in novel environments, the pose estimators often show degraded performance on out-of-distribution data (caused by e.g. illumination differences). Annotating target-domain real images for training is tedious, and limits the potential for autonomous operations. We propose to collect images during robot navigation, exploit the object SLAM estimates to pseudo-label the data, and fine-tune the pose estimator.

As depicted in Fig. 1, we develop a SLAM-supported self-training procedure for RGB-image-based object pose estimators. During navigation, the robot collects images and deploys a pre-trained model to infer object poses in the scene. Combining the pose estimates with noisy robot state measurements from on-board odometric sensors (camera, IMU, lidar, etc.), a pose graph optimization (PGO) problem is formulated to optimize the camera (i.e. robot) and object poses. We leverage the globally consistent state estimates to pseudo-label the images, generating new training data to fine-tune the initial model, and mitigate the domain gap in object pose prediction without human supervision.

A major challenge in this procedure is the difficulty of modeling the uncertainty of the learning-based pose measurements [3]. In particular, it is difficult to specify a priori an appropriate (Gaussian) noise model for them as typically required in PGO. For this reason, rather than fixing a potentially poor choice of covariance model, we allow the uncertainty model to change as part of the optimization process. Similar approaches have been explored previously in the context of robust SLAM (e.g. [4], [5]). However, our joint optimization formulation permits a straightforward alternating minimization procedure, where the optimal covariances are determined analytically and component-wise, allowing us to fit a richer class of noise models.

We tested our method with the deep object pose estimator (DOPE) [6] on the YCB video (YCB-v) dataset [7] and with real robot experiments. It can generate high-quality pseudo labels with average pixel error less than 3% of the image width. The DOPE estimators, after self-training, make respectively 34.3% and 17.8% more accurate predictions, and 25.2% and 25.7% fewer outlier predictions in the two experiments.

In summary, our work has the following contributions:

- A SLAM-aided object pose annotation method that allows robots to generate multi-view consistent pseudo training data for object pose estimators.
- A robust PGO method with automatic component-wise covariance fitting that is competitive with existing robust M-estimators, which could be of independent interest.
- We demonstrate that our method can effectively mitigate the domain gap for 6D object pose estimators under outlier-corrupted predictions and real sensory noise in robot navigation.

II. RELATED WORK

A. Semi- and self-supervised 6D object pose estimation

Training object pose estimators, which are typically data-hungry, requires tremendous annotation efforts [1]. While manual pose-labeling is labor-intensive, image synthesis tools (e.g. [8]) can efficiently generate large-scale training datasets and achieve extreme realism. However, the image rendering process still faces challenges such as realistic sensor noise simulation [9].

Semi- and self-supervised methods exploit unlabeled real images to mitigate the lack of data. They typically train...
the pose prediction model on synthetic data in a supervised manner and improve its performance on real images by semi- or self-supervised learning [10]–[17]. Many recent methods leverage differentiable rendering to develop end-to-end self-supervised pose estimators, by encouraging similarity between real images and images rendered with estimated poses [12]–[15]. However, most of the methods are not leveraging the spatial-temporal continuity of the robot-collected image streams. Deng et al. [18], instead, collect RGB-D image streams with a robot arm during object manipulation for self-training. They use a pose initialization module to obtain accurate pose estimates in the first frame, and use motion priors from forward kinematics and a particle filter [19] to propagate the pose estimates forward in time.

We propose to collect and pseudo-label RGB images during robot navigation. Instead of frame-to-frame pose tracking, we directly estimate the 3D scene geometry which requires no accurate first-frame object poses or high-precision motion priors. We use only RGB cameras and optionally other odometric sensors, which are mostly available on a robot platform, for data collection and self-training.

B. Multi-view object-based perception

Integrating information from multiple viewpoints can help correct the inaccurate and missed predictions by single-frame object perception models. For example, Pillai et al. [20] developed a SLAM-aware object localization system. They leverage visual SLAM estimates, consisting of camera poses and feature locations, to support the object detection task, leading to strong improvements over single-frame methods.

Beyond that, multi-view object perception models can provide noisy supervision to train single-frame methods. Mitash et al. [21] conduct multi-view pose estimation to pseudo-label real images for self-training of an object detector. Shugurov et al. [22] leverage relative camera poses from 2 or 4 views to refine object pose estimates and fine-tune the estimator networks. Nava et al. [23] exploit noisy state estimates to assist self-supervised learning of spatial perception models.

Inspired by these works, we propose to use SLAM-aided object pose estimation to generate training data for semi-supervised learning of object pose estimators.

C. Automatic covariance tuning for robust SLAM

Obtaining robust SLAM estimates is critical for the success of our self-training method. In PGO, the selection of measurement noise covariances relies mostly on empirical estimates of noise statistics. In the presence of outlier-prone measurements (e.g. learning-based) whose uncertainty is hard to quantify, it is infeasible to fix a (Gaussian) noise model for PGO. Adaptive covariance tuning methods (e.g. [24], [25]) have been developed to concurrently estimate the SLAM variables and noise statistics. The adaptive noise modeling technique in general brings about more accurate SLAM estimates.

The robust M-estimators are widely applied to deal with outlier-corrupted measurements (e.g. [4], [5], [26]). Minimizing their robust cost functions with the iteratively re-weighted least squares (IRLS) method, the measurement contributions are re-weighted at each step based on their Mahalanobis distances. This down-weights the influence of outliers and is de facto re-scaling the covariance uniformly [4]. Pfeifer et al. [5] propose to jointly optimize state variables and covariances, with the covariance components log-regularized. This method leads to a robust M-estimator, closed-form dynamic covariance estimation (cDCE), which can be solved with IRLS. It is more flexible than existing M-estimators since it allows the noise components to be re-weighted differently.

2834
Our joint optimization formulation is similar, with co-
v covariances L1 regularized, and also permits closed-form
component-wise covariance update. Instead of IRLS, we
follow an alternating minimization procedure, and eliminate
the influence of outliers based on the χ² test.

III. METHODOLOGY

A. Object SLAM via pose graph optimization

In our method, the object pose estimator is initially trained
on (typically synthetic) labeled image data \( D^i = \{ I^i, z^i, y^i \} \),
where \( I \) and \( z \) denote RGB images and the ground truth
6DoF object poses (w.r.t camera), and \( y \) represents the
ground truth labels. We can map \( z \) to \( y \) via an estimator-
dependent function \( y = \pi(z) \). In this paper, we work
with the DOPE [6] estimator\(^1\), for which \( y \) is the pixel locations
for the projected object 3D bounding box, and \( \pi(\cdot) \) is the perspective projection function. We apply the initial estimator
to make object pose predictions as the robot explores its
environment.

During navigation, the robot collects a number of un-
labeled RGB images \( D^n = \{ I^n \} \). The estimator makes
noisy object pose measurements (i.e. network predictions)
\( Z = \{ z_k \in SE(3) \}_{k=1}^{K} \) on \( I^n \). Combining \( Z \) and camera
odomometric measurements \( U = \{ u_k \in SE(3) \}_{k=1}^{K} \) obtained
with on-board sensors, a pose graph can be established as
\( G = \{ V, F, E \} \) (see Fig. 1(c)). \( V \) denotes the latent variables
incorporating camera poses \( X = \{ x_t \in SE(3) \}_{t=0}^{T} \) and
object landmark poses (w.r.t world) \( L = \{ \ell_j \in SE(3) \}_{j=1}^{J} \). Assuming
all the measurements are independent and the measurement-object correspondences are known, we can
decompose the joint likelihood \( p(Z, U \mid X, L) \) into factors
\( F = \{ F_U, F_Z \} \), consisting of odometry factors \( f_t \in F_U \)
and object pose measurement factors \( f_k \in F_Z \). Each factor
describes the likelihood of a measurement given certain
variable assignments. Edges \( E \) represent the factorization
structure of the likelihood distribution:

*\[
p(Z, U \mid X, L) = \prod_t f_t \prod_k f_k = \prod_t p(u_t \mid x_{t-1}, x_t) \prod_k p(z_k \mid x_t, \ell_j) \tag{1}
\]*

We assume the measurement noises are zero-mean, normally-
distributed \( \mathcal{N}(0, \Sigma_k) \), where \( \Sigma_k \) is the noise covariance matrix for measurement \( z_k \). The optimal assignments of the
variables can be obtained by solving the maximum likelihood estimation problem:

*\[
\hat{X}, \hat{L} = \arg\max_{X, L} p(Z, U \mid X, L) = \arg\min_{X, L} - \log p(Z, U \mid X, L)
= \arg\min_{X, L} \sum_t \| \log(u_t^{-1} x_{t-1} x_t) \|_{\Sigma_t}^2 + \sum_k \| \log(z_k^{-1} x_t \ell_j) \|_{\Sigma_k}^2 \tag{2}
\]*

\(^1\)Our method is not restricted to using a particular pose estimator.

where \( \log: SE(3) \rightarrow \mathbb{R}^6 \) is the logarithmic map on the SE(3)
group, and \( \| \cdot \|_{\Sigma} \) is the Mahalanobis distance. We solve the
least-squares optimization (2) with self-tuned covariances \( \Sigma_k \)
(see Sec. III.C) to obtain robust SLAM estimates \( \{ \hat{x}_t \}, \{ \hat{\ell}_j \} \).

B. Hybrid pseudo-labeling

Based on the optimal states, we can: (1) identify inliers \( Z^m \)
from object pose measurements \( Z \); (2) compute optimized
object-to-camera poses \( \hat{z}_{ij} = x_t^{-1} \ell_j \). We compare and
combine \( Z^m \) and \( \{ \hat{z}_{ij} \} \) to obtain the pseudo ground truth poses \( \bar{z}^m \) for images in \( D^n \) (see Fig. 1(e)). We refer to
the pseudo-labeling method as Hybrid labeling, because the pseudo
labels are derived from two sources.

First, we identify inlier pose measurements \( \bar{z}^m_k \in Z^m \subset Z \)
based on the \( \chi^2 \) test (Alg. 1 line 5):

*\[
\| \log(x_t^{-1} \hat{x}_t^{-1} \hat{\ell}_j) \|^2_{\Sigma_k} < \chi^2_{0.95} \tag{3}
\]*

where \( \chi^2_{0.95} \) is the critical \( \chi^2 \) value for 6 DoF at confidence
level 0.95.

Second, we directly use optimized object-to-camera poses
as pseudo ground truth poses. For the image collected at time
\( t \), the optimized object-to-camera pose for object number \( j \)
is computed as \( \hat{z}_{ij} = x_t^{-1} \ell_j \). However, the inlier predictions
or optimized poses can be noisy and may deviate visually from
the target objects on the images and hurt training.

Therefore, we employ a pose evaluation module, as proposed in [18], to quantify the visual coherence between
pseudo labels and images. Each RGB image is compared with
its corresponding rendered image based on the pseudo
ground truth pose \( \bar{z}^m \). The regions of interest (RoIs) on
the two images are passed into a pre-trained auto-encoder
from PoseRBPF [19], and the cosine similarity score of
their feature embeddings is computed. We abstract the score
calculation process as a function \( \phi: SE(3) \rightarrow [0, 1] \), where
the dependence on object, camera and image information is
made implicit.

We compute the PGO-computed pose’s score \( \phi(\hat{z}) \) and
the inlier score \( \phi(\bar{z}^m) \) for every target object on unlabeled
images, and pick the pose with a higher score as the
pseudo ground truth pose \( \bar{z}^m \). The regions of interest (RoIs) on
the two images are passed into a pre-trained auto-encoder
from PoseRBPF [19], and the cosine similarity score of
their feature embeddings is computed. We abstract the score
calculation process as a function \( \phi: SE(3) \rightarrow [0, 1] \), where
the dependence on object, camera and image information is
made implicit.

We compute the PGO-computed pose’s score \( \phi(\hat{z}) \) and
the inlier score \( \phi(\bar{z}^m) \) for every target object on unlabeled
images, and pick the pose with a higher score as the
pseudo ground truth pose \( \bar{z}^m \). The regions of interest (RoIs) on
the two images are passed into a pre-trained auto-encoder
from PoseRBPF [19], and the cosine similarity score of
their feature embeddings is computed. We abstract the score
calculation process as a function \( \phi: SE(3) \rightarrow [0, 1] \), where
the dependence on object, camera and image information is
made implicit.

C. Automatic covariance tuning by alternating minimization

Since the object pose measurements \( Z \) are derived from
learning-based models, it is difficult to a priori fix a (Gaus-
sian) uncertainty model for them as typically required in
PGO. We propose to automate the covariance tuning process
by incorporating the noise covariances as variables into a
joint optimization procedure. Our formulation, as we show, leads to a simple and robust PGO method, that allows us to alternate between PGO and determining in closed form the optimal noise models.

For simplicity, we rewrite the PGO cost function in (2) as:

$$\mathcal{L} = \sum_k \| e_k \|_{\Sigma_k}^2 + \sum_t \| e_t \|_{\Sigma_t}^2$$

(4)

where $e_k$ and $e_t$ are the residual error vectors.

Given that the odometry measurements $U$ are typically more reliable, we only update the object pose noise covariances $\Sigma_k$ at the covariance optimization step. Each time after optimizing SLAM variables, we solve for the per-measurement noise models $\Sigma_k$ that can further minimize the optimization loss, and re-solve the SLAM variables with them. To avoid the trivial solution of $\Sigma_k \rightarrow \infty$, we apply L1 regularization on the covariance matrices. The loss function for the joint optimization is:

$$\mathcal{L}_{\text{joint}} = \sum_k \| e_k \|_{\Sigma_k}^2 + \lambda \sum_k \| \Sigma_k \|_1 + \sum_t \| e_t \|_{\Sigma_t}^2$$

(5)

For simplicity, we assume the covariance matrices are diagonal, i.e. $\Sigma_k = \text{diag}(\sigma_{k1}^2, \cdots, \sigma_{k6}^2)$, and the joint loss reduces to:

$$\mathcal{L}_{\text{joint}} = \sum_{k,j=1}^{6} \frac{e^2_{k,j}}{\sigma_{k,j}^2} + \lambda \sum_{k,j=1}^{6} \sigma_{k,j}^2 + \sum_t \| e_t \|_{\Sigma_t}^2$$

(6)

where $e_{k,j}$ is the $j$th entry of the residual error $e_k$.

For fixed $e_t$ and $e_{k,j}$, optimizing (6) w.r.t $\sigma_{k,j}^2$ amounts to minimizing a convex objective over the non-negative reals. Moreover, we know $\sigma_{k,j}^2 = 0$ is never a minimizer for $\mathcal{L}_{\text{joint}}$. Thus, a global optimal solution is obtained when:

$$\frac{\partial \mathcal{L}_{\text{joint}}}{\partial \sigma_{k,j}^2} = -\frac{e^2_{k,j}}{\sigma_{k,j}^2} + \lambda = 0$$

(7)

evaluating which we obtain the extremum on $\mathcal{L}_{\text{joint}}$:

$$\sigma_{k,j}^2 = \frac{|e_{k,j}|}{\sqrt{\lambda}} \quad j = 1, \cdots, 6$$

(8)

With $\partial^2 \mathcal{L}_{\text{joint}}/\partial \sigma_{k,j}^2 \geq 0$ being valid for all $\sigma_{k,j}^2 > 0$, we can confirm that this extremum is a global minimum.

Therefore we can express the covariance update rule at iteration $i$ as:

$$\Sigma_{k}^{(i)} = \text{diag}(\lambda' |e_k^{(i)}|)$$

(9)

where we define $\lambda' = 1/\sqrt{\lambda}$ for the convenience of $\lambda$ tuning.

Since the covariance optimization step admits a closed form solution, the joint optimization reduces to “iteratively solving PGO with re-computed covariances”. The selection of all the noise models reduces to tuning $\lambda$. According to our experiments, it’s feasible to set $\lambda$ as a constant for consistent performance in different cases.

The algorithm is summarized in Alg. 1. We use the Levenberg-Marquardt (LM) algorithm to solve the PGO with Gaussian noise models (line 2). For better performance, we also identify outliers at each iteration using the $\chi^2$ test.

\[\chi^2 = 10 \text{ (i.e. } \lambda = 0.01\text{) for all the tests.}\]

The update rule (9) reveals our implicit assumption that a high-residual measurement component is likely from a high-variance normal distribution. The recomputed covariances are typically down-weight high-residual measurements, which is in spirit similar to robust M-estimation. We show in Appendix B that, with isotropic noise models, our method reduces to using the L1 robust kernel.

However, the robust M-estimation is typically solved with the IRLS method, where the measurement losses are re-weighted based on the Mahalanobis distance $\| e_k \|_{\Sigma_k}$ (see (17)). In comparison, our method re-weights the measurement losses component-wise using residual error components. This enables us to fit a richer class of noise models for 6DoF PGO, in that different components often follow different noise characteristics.

### IV. Experiments

Our method is tested with (1) the YCB-v dataset and (2) a self-collected real robot experiment. On the YCB-v dataset (Sec. IV.A), we leverage the image streams in the training sets to solve per-video PGOs for self-training. We also conduct ablation experiments to compare the Hybrid labeling method against another two baseline methods. In the real robot experiment (Sec. IV. B), we apply the method on longer sequences, circumnavigating selected objects, and test our method’s robustness to challenges in ground robot navigation, e.g. increased motion blur.

Our method is implemented in Python. We use the NVISII toolkit [8] to generate synthetic image data. The training and inference programs are adapted from the code available in the DOPE GitHub [6]. Every network is initially trained for 60 epochs and fine-tuned for 20 epochs, with a batchsize of 64 and learning rate of 0.0001. We solve PGO problems based on the GTSAM library [27]. We run the synthetic data generation and training programs with 2 NVIDIA Volta V100 GPUs and other code on an Intel Core i7-9850H@2.6GHz CPU and an NVIDIA Quadro RTX 3000 GPU.
TABLE I. Comparison of robust PGO methods via pseudo label accuracy on YCB-v sequences. Column 1-7 are median (pixel) errors in PGO-generated pseudo labels $y_{ij}$ for the first 7 (out of 20) videos. Column 8 (#best) is the number of sequences on which a method achieves the lowest error.

| Method         | 001_cracker_box | 003_cracker_box | 004_sugar_box | 009_potted_meat | 010_potted_meat | 011_potted_meat | 014_cracker_box | #best |
|----------------|-----------------|-----------------|----------------|-----------------|-----------------|-----------------|-----------------|-------|
| LM             | 62.3            | 58.7            | 69.4           | 57.6            | 110.1           | 101.6           | 0               | 6     |
| Cauchy         | 62.8            | 59.2            | 71.0           | 57.6            | 110.1           | 101.6           | 0               | 6     |
| Huber          | 55.4            | 54.9            | 64.2           | 52.5            | 57.0            | 1               | 1               | 6     |
| Geman-McClure  | 115.4           | 146.4           | 155.0          | 94.6            | 171.4           | 1                | 2               | 6     |
| cDCE [3]       | 28.7            | 25.4            | 30.5           | 25.2            | 58.9            | 4               | 1               | 6     |
| ACT/estimators | 15.7            | 12.0            | 9.4            | 12.6            | 20.8            | 32.0            | 5               | 6     |

| Method         | 001_mo天文     | 002_box         | 003_box         | 004_potted_meat | 005_potted_meat | 006_potted_meat | 007_sugar_box   | #best |
|----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-------|
| LM             | 29.9            | 25.1            | 28.7            | 23.8            | 75.1            | 73.3            | 6               | 6     |
| Cauchy         | 27.6            | 23.8            | 23.8            | 17.9            | 104.4           | 6.4             | 1               | 7     |
| Huber          | 11.7            | 12.8            | 15.5            | 25.4            | 71.1            | 6               | 1               | 7     |
| Geman-McClure  | 14.0            | 11.8            | 14.0            | 14.3            | 106.4           | 6.4             | 1               | 7     |
| cDCE [3]       | 12.3            | 12.0            | 13.1            | 14.9            | 20.7            | 7.2             | 5               | 6     |
| ACT/estimators | 8.2             | 5.9             | 5.4             | 12.1            | 18.0            | 6.0             | 1               | 7     |

4 We implemented cDCE via replacing (8) with equation (16) in [5].

A. YCB video experiment

We tested our method with the DOPE estimators [6] for three YCB objects: 003_cracker_box, 004_sugar_box and 010_potted_meat_can. Each object appears in 20 YCB videos (training + testing). They have respectively 26689, 22528 and 27050 training images (dashed lines in Fig. 2(a)) from 17, 15, and 17 training videos.

We use 60k NVISII-generated synthetic image data $D^i$ to train initial DOPE estimators for the 3 objects. The models are applied to infer object poses on the YCB-v images. We employ the ORB-SLAM3 [28] RGB-D module (w/o loop closing) to obtain camera odometry on these videos.

Combining the measurements, we solve per-video PGOs to refine object-to-camera poses for pseudo-labeling. We initialize the camera poses with the odometry chain, object poses with average pose predictions, and covariances with $\Sigma_k = 0.1\Sigma$, $\Sigma_0 = 0.01\Sigma$. We apply different robust optimization methods to solve all 60 PGO problems (see Tab. I (56)). For comparison purposes only, we compute pseudo labels for all the YCB-v images directly from the optimal states, i.e. $y_{ij} = \pi(\hat{x}_i^{-1}\hat{y}_j)$, and compare the methods via label errors, i.e. how much the projected object bounding boxes deviate from the ground truth. With the component-wise noise modelling and mitigation of outlier corruption, our method (ACT) achieves the lowest errors in much more videos (see Tab. I Col. 8). It performs stably across sequences based on a fixed initial guess and a constant regularization coefficient $\lambda$. Therefore, we elect to pseudo-label the YCB-v training images based on the results by ACT.

To evaluate different modules, we compare our Hybrid method with another two baseline labeling methods: Inlier and PoseEval. Inlier uses the PGO results only as an inlier measurement filter. The raw pose predictions that agree geometrically with other measurements are selected for labeling, i.e. $\hat{z}^{ps} = z^m$. PoseEval extracts visually coherent pose predictions by thresholding the similarity scores from the pose evaluation module, i.e. $\hat{z}^{ps} = \{z_k | \phi(z_k) > \hat{s}^{ps}\}$.

The Hybrid method ensures spatial and visual consistency of the pseudo-labeled data. For a certain target object in an image, we compare the pose evaluation score for the inlier prediction $\phi(z_k^*)$ (if available) with that of the optimized object pose $\phi(\hat{z})$. The higher scorer, if beyond a threshold, is picked for labeling. Thus, the Hybrid pseudo ground truth poses can be expressed as: $\hat{z}^{ps} = \{z_k | \phi(z_k) > \phi(z_k^*) \land \phi(z) > \phi(z^*) \land \phi(z_k) > \hat{s}^{ps}\} \cup \{z_k | \phi(z_k) > \phi(\hat{z}) \land \phi(z_k) > \hat{s}^{ps}\}$, where the thresholds satisfy $s^{ps} > s^{in}$ because the PGO-generated labels, not directly derived from RGB images, are prone to misalignment. Thus, the Hybrid pseudo-labeled data $D^{ps}$ consists of high-score inliers, PGO-generated easy examples, and hard examples (on which the initial estimator fails). The 3 components are colored differently on Hybrid bars in Fig. 2(a). For the Hybrid and Inlier modes, we also exclude YCB-v sequences with measurement outlier rates higher than 20% from pseudo-labeling.

The statistics of the pseudo-labeled data are reported in Fig. 2. The Hybrid and Inlier data are in general very accurate (average label errors $< 3\%$ of image width), with the $\chi^2$ test being an effective inlier filter. But the PoseEval data, whatever their sizes, are more noisy and outlier-corrupted, so we cannot rely only on the pose evaluation test to generate outlier-free labels.

We evaluate different self-training methods, using the estimator performance gain, on the YCB-v test sets. We adopt the average distance (ADD) metric [30], and present the accuracy-threshold curves in Fig. 4. All the methods achieve considerable improvements over the initial model, indicating that the real image data, although noisy and outlier-corrupted, are highly effective for decreasing the domain gap, as also reported in [12]–[15], [17]. But they still have large performance gaps from the models trained by means of supervised learning, due to noises and the limited data size. Our Hybrid method consistently outperforms other
baselines, even though its data have similar statistics with the Inlier data. We thus infer that the performance gain mainly comes from the presence of hard examples.

We also present the performance gain qualitatively in Fig. 3, and the outlier prediction reduction after self-training in Tab. II. We observe that the per-frame pose predictions become more consistent across frames and the outlier predictions are minimized (-25.2%) after self-training, even if the initial predictions are outlier-contaminated.

Further, we examine whether the fine-tuned estimator brings about improved object SLAM accuracy. The YCB-v test sequences 0049, 0059 are selected for evaluation (since they have 2 out of the 3 selected YCB objects). We solve the PGOs with the LM algorithm, using object pose predictions before and after the estimator is fine-tuned. The estimation errors are reported in Tab. III. We observe that camera and object pose estimations are consistently more accurate after model self-training.

TABLE II. The outlier prediction rates on the YCB-v test sets before and after self-training. We use the $\chi^2$ test to identify outliers from all the object pose predictions.

|                | % Outliers Before | % Outliers After |
|----------------|-------------------|------------------|
| 003_cracker_box| 58.43             | 12.07            |
| 004_sugar_box  | 21.76             | 0.27             |
| 010_potted_meat_can | 15.06 | 2.29             |
| Total          | 29.07             | 3.88             |

B. Real robot experiment

As illustrated in Fig. 5, we control a Jackal robot [31] to circle around two target objects: 003_cracker_box and 010_potted_meat_can, and collect stereo RGB images from a ZED2 camera [32]. For each object, two ~4 min. long sequences are recorded, one for self-training and the other for testing. We obtain the ground truth camera trajectory from a Vicon MoCap system [33] and the ground truth object poses from AprilTag detections [34]. The camera odometry is computed with the SVO2 stereo module [35]. We infer the object poses from the left camera RGB images using the same initial estimator as in Sec. IV.A.

Similar to the YCB-v experiment, we solve PGOs and pseudo-label the left camera images with the Hybrid method. For the two objects, 648 (out of 3120) and 950 (out of 2657) images are pseudo-labeled, in which 1.5% and 2.8% are hard examples. After fine-tuning, we evaluate the estimator model on both self-collected test sequences and the YCB-v test sets. On the self-collected test sequences, we adopt the reprojection error for the object 3D bounding box as the evaluation metric. The accuracy-threshold curves are presented in Fig 6. The AUCs for the curves in both tests are reported in Tab. IV.

We observe that the self-trained estimator makes considerably more (+17.75% on average) accurate predictions in the same experimental environment. More interestingly, in


**Fig. 6.** Quantitative evaluation of our method on the test sets of the real robot experiment: Reprojection error accuracy-threshold curves before and after self-training. (Image size: 376x672.)

another unseen real-world domain, YCB-v test scenes, the self-trained model also shows enhanced prediction accuracy (+4.8 on average). We infer that this is because the domain gap between real environments is less than the sim2real gap.

**TABLE IV.** Quantitative evaluation of our self-training method on the real robot experiment: the area under curve (AUC) values (%) for the accuracy-threshold curves on the self-collected and the YCB-v test sets before and after self-training.

|                         | Reproj error AUC | 003_cracker_box | 010_potted_meat can |
|-------------------------|------------------|-----------------|---------------------|
|                         | Before           | After           | Before              | After              |
|                         | 55.0             | 63.9            | 40.6                | 70.2               |

|                         | ADD AUC          | 003_cracker_box | 010_potted_meat can |
|-------------------------|------------------|-----------------|---------------------|
|                         | Before           | After           | Before              | After              |
|                         | 8.0              | 15.2            | 42.3                | 44.7               |

Similar to the YCB-v experiment, we further evaluate the SLAM performance gain after self-training. We solve the PGOs on the self-collected test sequences and the YCB-v test set 0059, with the initial and the self-trained models’ predictions. As reported in Tab. V, the trajectory estimation errors are reduced in the same test environment and in another unseen YCB-v test domain.

**TABLE V.** Object SLAM accuracy on the self-collected test sequences and the YCB-v test set 0059 before and after self-training: absolute trajectory errors (ATE).

| ATE (m)                  | 003_cracker_box test | 010_potted_meat can test | YCB-v test set 0059 |
|--------------------------|----------------------|--------------------------|---------------------|
| Before                   | 0.046                | 0.721                    | 0.618               |
| After                    | 0.046                | 0.721                    | 0.576               |

**V. CONCLUSIONS**

We present a SLAM-aided self-training method for domain adaptation of 6D object pose estimators. We exploit robust PGO results to generate multi-view consistent pseudo labels on robot-collected images and fine-tune the estimator models. We propose an easy-to-implement robust PGO method, that can automatically tune the pose prediction covariances component-wise. We evaluate our method on the YCB-v dataset and with real robot experiments. The method can mine high-quality pseudo-labeled data from noisy and outlier-corrupted pose predictions. After self-training, the pose estimators (initially trained on synthetic data) show considerably more accurate and consistent performance in the test domain, which also boosts the object SLAM accuracy.

**APPENDIX**

A. Proof for monotonic improvements in the joint loss

We prove that our automatic covariance tuning method can monotonically reduce the joint loss $L_{\text{joint}}(\mathcal{V}, \Sigma_k)$ as defined in (5). At iteration $i$, we have the variable assignments $\mathcal{V}^{(i)}$ and the noise covariances $\Sigma_k^{(i)}$ from iteration $i-1$. Solving the PGO, the LM algorithm ensures $L(\mathcal{V}^{(i)}) \geq L(\mathcal{V}^{(i-1)})$. Since the extra regularization term in $L_{\text{joint}}$ is not a function of $\mathcal{V}$, we can have:

$$L_{\text{joint}}(\mathcal{V}^{(i)}, \Sigma_k^{(i)}) \leq L_{\text{joint}}(\mathcal{V}^{(i-1)}, \Sigma_k^{(i-1)})$$  \hspace{1cm} (10)

We have also shown after (8) that $\Sigma_k = \text{diag}(|e_k|/\sqrt{\lambda})$ is a global minimizer for $L_{\text{joint}}$. Thus, we have:

$$L_{\text{joint}}(\mathcal{V}^{(i)}, \Sigma_k^{(i)}) \leq L_{\text{joint}}(\mathcal{V}^{(i)}, \Sigma_k^{(i-1)})$$  \hspace{1cm} (11)

Combining the two inequalities, we obtain:

$$L_{\text{joint}}(\mathcal{V}^{(i)}, \Sigma_k^{(i)}) \leq L_{\text{joint}}(\mathcal{V}^{(i-1)}, \Sigma_k^{(i-1)})$$  \hspace{1cm} (12)

With this being valid for all iterations, we can have the chain of inequalities:

$$L_{\text{joint}}^{(1)} \geq L_{\text{joint}}^{(1)} \geq \cdots \geq L_{\text{joint}}^{(N)}$$  \hspace{1cm} (13)

which completes the proof.

In our implementation, we eliminate outliers’ influence by setting their $\Sigma_k$ to a large value (Alg. 1 line 8), which may violate this monotonic improvement property. We choose to freeze the losses of outliers in $L_{\text{joint}}$ till the end of optimization to resolve the problem.

B. Reduction to L1 robust M-estimator

We show that our automatic covariance tuning method, as the noise models $\Sigma_k$ are assumed to be isotropic, is equivalent to using the L1 robust M-estimator. With the isotropic noises, i.e. $\Sigma_k = \sigma_k I$, the joint loss in (5) reduces to:

$$L_{\text{joint}} = \sum_k \frac{1}{\sigma_k^2} \|e_k\|_2^2 + 6\lambda \sum_k \sigma_k^2 + \sum_t \|e_t\|^2_{\Sigma_t}$$  \hspace{1cm} (14)

Evaluating $\partial L_{\text{joint}}/\partial \sigma_k^2 = 0$ yields the new update rule:

$$\Sigma_k^{(i)} = \frac{1}{\sqrt{\mathcal{V}^{(i)}}} |e_k^{(i)}| I$$  \hspace{1cm} (15)

On the other hand, we can apply the L1 robust loss for the object pose measurement factors $f_k \in F_Z$ to minimize the PGO loss (4). The robust PGO cost can be expressed as:

$$L_{\text{robust}} = \sum_k \rho(||e_k||_{\Sigma_k}) + \sum_t \|e_t\|_{\Sigma_t}^2$$  \hspace{1cm} (16)

where $\rho(\cdot) = |\cdot|$ is the L1 robust kernel. The robust cost is typically minimized with the IRLS method, by matching the gradients of (16) locally with a sequence of weighted least squares problems. The local least squares formulation at iteration $i$ can be expressed as:

$$L_{\text{IRLS}} = \sum_k w(||e_k^{(i)}||_{\Sigma_k})||e_k||_{\Sigma_k}^2 + \sum_t \|e_t\|_{\Sigma_t}^2$$  \hspace{1cm} (17)
where \( w(\cdot) = 1/|\cdot| \) is the weight function. Under the isotropic noise assumption, i.e. \( \Sigma_k = \sigma_k I \), we can absorb the weight function into the covariance matrix and rewrite (17) as:

\[
\mathcal{L}_{RLS} = \sum_k \| e_k \|^2_{\Sigma_k^{-1}(\cdot)} + \sum_t \| e_t \|^2_{\Sigma_t},
\]

where the covariance matrix is de facto re-scaled iteratively by:

\[
\Sigma_k^{(i)} = \sigma_k |e_k^{(i)}| I
\]

Matching (15) with (19), we can see as \( \frac{1}{\sqrt{\sigma_k}} = \sigma_k \), the two methods are in theory equivalent.\(^9\)

ACKNOWLEDGMENT

The authors acknowledge Jonathan Tremblay and other NVIDIA developers for providing consultation on training DOPE networks and generating synthetic data. The authors also acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for HPC resources that have contributed to the results reported within this paper.

REFERENCES

[1] C. Sahin, G. Garcia-Hernando, J. Sock, and T.-K. Kim, “A review on object pose recovery: from 3d bounding box detectors to full 6d pose estimators,” *Image and Vision Computing*, vol. 96, p. 103898, 2020.

[2] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, “Slam++): Simultaneous localisation and mapping at the level of objects,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 1352–1359.

[3] M. Abdar, F. Pourpanah, S. Hussain, D. Rezzagadeh, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, et al., “A review of uncertainty quantification in deep learning: Techniques, applications and challenges,” *Information Fusion*, vol. 76, pp. 243–297, 2021.

[4] P. Aragwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, “Robust map optimization using dynamic covariance scaling,” in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 62–69.

[5] T. Pfeifer, S. Lange, and P. Protzel, “Dynamic covariance estimation—a parameter free approach to robust sensor fusion,” in *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2017, pp. 359–365.

[6] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, “Deep object pose estimation for semantic robotic grasping of household objects,” *arXiv preprint arXiv:1809.10790*, 2018.

[7] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, “Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes,” *arXiv preprint arXiv:1711.00599*, 2017.

[8] N. Morrical, J. Tremblay, S. Birchfield, and I. Wald, “NVISIV: Nvdis scene imaging interface,” 2020, https://github.com/owl-project/NVISIV.

[9] S. I. Nikolenko, *Synthetic data for deep learning*. Springer, 2021, vol. 174.

[10] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, “Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge,” in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 1386–1383.

[11] Z. Li, Y. Hu, M. Salzmann, and X. Ji, “Robust rgb-based 6-dof pose estimation without real pose annotations,” *arXiv preprint arXiv:2008.08394*, 2020.

[12] G. Wang, F. Manhardt, J. Shao, X. Ji, N. Navab, and F. Tombari, “Self6d: Self-supervised monocular 6d object pose estimation,” in *European Conference on Computer Vision*. Springer, 2020, pp. 108–125.

\(^9\)\( \sigma_k = \text{const.} \) in the context of robust M-estimation.