Supplementary Material

Kamil Dreczkowski[1] and Edward Johns[1]

I. Fusing *SE*(3) Poses

In this project, we use the framework proposed by Barfoot et al. [1] to parameterise uncertainties associated with *SE*(3) poses and for fusing multiple poses into a single estimate. We represent poses as 4×4 transformation matrices

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \in SE(3)$$

where $\boldsymbol{R} \in SO(3)$ is a rotation matrix and $\boldsymbol{t} \in \mathbb{R}^3$ is a translational vector. Since $\boldsymbol{T} \in SE(3)$, it is not straightforward to associate uncertainties with poses or to fuse multiple estimates. For example, one would not be able to model uncertainties using the usual approach of additive uncertainty as poses are not members of a *vector space*. In [1], a random variable for *SE*(3) is defined according to

$$\boldsymbol{T} = \exp(\boldsymbol{\xi}^\wedge)\bar{\boldsymbol{T}}$$

where $\bar{\boldsymbol{T}} \in SE(3)$ is a "large" noise-free value and $\boldsymbol{\xi} \in \mathbb{R}^6$ is a "small" noisy perturbation. The $^\wedge$ operator turns $\boldsymbol{\xi}$ into a 4×4 member of the *Lie algebra se*(3) according to

$$\boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \boldsymbol{0}^T & 0 \end{bmatrix}$$

where $\boldsymbol{\rho}, \boldsymbol{\phi} \in \mathbb{R}^3$ and $^\wedge$ also turns $\boldsymbol{\phi}$ into a member of the *Lie algebra so*(3):

$$\boldsymbol{\phi}^\wedge = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}$$

The operator $^\vee$ is the inverse of $^\wedge$. The small perturbation variable $\boldsymbol{\xi}$ is assumed to be distributed according to a zero-mean Gaussian $p(\boldsymbol{\xi}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ where $\boldsymbol{\Sigma} \in \mathbb{R}^{6\times 6}$ is a covariance matrix. This covariance matrix parameterises the uncertainty in a $SE(3)$ pose.

To fuse multiple pose estimates into a single pose, Barfoot et al. [1] proposes to use the iterative Gauss-Newton like algorithm shown in algorithm 1. This algorithm is initialised with K *SE*(3) estimates and their associated uncertainties

$$\{\bar{\boldsymbol{T}}_1, \boldsymbol{\Sigma}_1\}, \{\bar{\boldsymbol{T}}_2, \boldsymbol{\Sigma}_2\}, ..., \{\bar{\boldsymbol{T}}_K, \boldsymbol{\Sigma}_K\}$$

and sets the initial estimate of the fused pose to $\bar{\boldsymbol{T}} = \boldsymbol{T}_1$. During each iteration of the algorithm, it constructs a linear system of equations $\boldsymbol{A}\boldsymbol{\xi} = \boldsymbol{b}$ and solves for $\boldsymbol{\xi}$ that is then used to update the current estimate of the fused measurement according to $\bar{\boldsymbol{T}} \leftarrow \exp(\boldsymbol{\xi}^\wedge)\bar{\boldsymbol{T}}$. In this algorithm, $\ln(\cdot)$ is the inverse of the exponential map of the *SE*(3) group and $\boldsymbol{\mathcal{J}}_k^{-1}$ is the inverse of the Jacobian for the *SE*(3) group for the transformation $\boldsymbol{T}_k$ (see [1] for more details).

Algorithm 1: Algorithm for fusing K *SE*(3) estimates with associated uncertainties

```
Input: $K$ $SE(3)$ poses with associated uncertainties,
       $\{\bar{\boldsymbol{T}}_1, \boldsymbol{\Sigma}_1\}, ..., \{\bar{\boldsymbol{T}}_K, \boldsymbol{\Sigma}_K\}$
Output: A single estimate $\bar{\boldsymbol{T}}$ with an associated
        uncertainty $\boldsymbol{\Sigma}$
Set $\bar{\boldsymbol{T}} = \boldsymbol{T}_1$
for $iter = 1, ..., max\_iter$ do
    $\boldsymbol{A} = \boldsymbol{0} \in \mathbb{R}^{6\times 6}$
    $\boldsymbol{b} = \boldsymbol{0} \in \mathbb{R}^6$
    for $k = 1, ..., K$ do
        $\boldsymbol{\xi}_k \leftarrow \ln(\bar{\boldsymbol{T}}\boldsymbol{T}_k^{-1})^\vee$
        Compute $\boldsymbol{\mathcal{J}}_k^{-1}$, using equation 103 from [1]
        $\boldsymbol{A} \leftarrow \boldsymbol{A} + \boldsymbol{\mathcal{J}}_k^{-T}\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mathcal{J}}_k^{-1}$
        $\boldsymbol{b} \leftarrow \boldsymbol{b} - \mathcal{J}_k^{-T}\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\xi}_k$
    Compute $\boldsymbol{\xi} = \boldsymbol{A}^{-1}\boldsymbol{b}$
    $\bar{\boldsymbol{T}} \leftarrow \exp(\boldsymbol{\xi}^\wedge)\bar{\boldsymbol{T}}$
$\boldsymbol{\Sigma} \leftarrow \boldsymbol{A}^{-1}$
```

There exist several methods for estimating the covariance matrix associated with a pose estimate. For example, Brossard et al. [2] propose an analytical expression for estimating this covariance matrix for point-to-plane ICP. This expression consists of two terms. The first term relates to initialisation uncertainty and accounts for wrong convergence and the lack of constraints in the input point clouds and is estimated by running ICP 12 times using different initialisations sampled from an initialisation distribution. The second term relates to sensor noise and is computed in closed form. As running ICP 12 times would increase its computation time by more than an order of magnitude, we have only experimented with using the second term to model uncertainties and have found that it only captures local uncertainty that does not reflect the accuracy of estimates. For this reason, we use the mean VSD estimate (MVE) introduced in section IV-A.2 of our main paper multiplied by the 6×6 identity matrix as a covariance matrix proxy. Although not theoretically grounded, the MVE is relatively cheap to compute and broadly captures the accuracy of an estimate.

Since the MVE serves as a scalar proxy to the full *SE*(3) uncertainty, it is not propagated through the individual dimensions of the camera transformation when transforming an estimate from one frame of reference of the camera to another in the "Sequential ICP experiment". Instead, we

[1] The Robot Learning Lab at Imperial College London kamil.dreczkowski15@imperial.ac.uk

propagate the entire scalar such that its magnitude is the same before and after the camera motion. Finally, we also use algorithm 1 to take a deterministic "*average*" of multiple $SE(3)$ estimates by setting the covariance matrix of all input poses to the 6×6 identity matrix.

II. Experimental Procedure

A. Implementation Details

All depth images of object were of resolution 480×640 and were rendered using the Python renderer from the BOP toolkit [3] using the camera intrinsic matrix

$$\boldsymbol{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 320 & 0 & 319.5 \\ 0 & 320 & 239.5 \\ 0 & 0 & 1 \end{bmatrix}$$

All ICP variants were based on the generic ICP algorithm (see Algorithm 1 in our main paper), were implemented in Python, and were run until convergence. NN variants were based on the Open3D [4] implementation of ICP, while projective variants were implemented to run on the GPU. For point selection, NN variants used all model points, and projective variants used all points from V_O (i.e. only the object geometry visible at ICP initialisation; see section III-A.1 of our main paper). For weighing correspondences, we use constant weights. Finally, for outlier rejection, we use a maximum distance between correspondences $\tau_{max} = 5$ cm and maximum angle between their normals $\theta_{max} = 45°$. Surface normals were estimated using the functionality of Open3D for NN variants and using four nearest neighbours for projective variants [5]. Segmentation masks were obtained directly from the renderer. Depth images were smoothed with a bilateral filter prior to ICP. We terminate any algorithm if the mean loss decreases by less than 0.1% between any two iterations.

B. Objects

We consider 20 ShapeNet [6] objects from 20 different object categories. Namely, these objects were: a calculator, cap, glasses, kettle, pen, plate, shoe, stapler, toaster, vase, bottle, bowl, camera, can, drinking glass, knife, light bulb, mug, phone and remote controller. These objects are representative of objects that a robot might encounter in household environments. We use the first 10 of these object (shown in the top row of figure 2 in our main paper) to tune the Dynamic Switching threshold. We then use the remaining 10 objects (shown in the bottom row of figure 2 in our main paper) to benchmark Hybrid ICP against other commonly used ICP algorithms.

C. Object Model representation

1) Noiseless object models: : Object CAD models are typically represented by triangular meshes. For algorithms that use NN data association, we represent object models by 8192 uniformly sampled points from a triangular mesh surface. For algorithms that use projective data association, we render a depth image from the triangular mesh and convert this depth image to a vertex map using the camera intrinsic matrix.

2) Noisy object models: : Noisy object models were obtained by integrating multiple depth images into a TSDF volume [7]. To obtain point clouds for NN ICP variants, we have first extract triangular meshes from the TSDF volumes using the marching cubes algorithm [8]. We have then uniformly sampled 8192 points from each of the meshes. For algorithms that use projective data association, we render depth images directly from the TSDF volumes with ray casting, and convert these depth images to vertex maps using the camera intrinsic matrix.

D. Sampling object poses

The algorithm used to sample an object pose in the camera frame is shown in algorithm 5. This algorithm begins by sampling a distance between the camera and the object, r, and by sampling a 3D position $\boldsymbol{x}$ on a unit sphere. It then sets the position of the camera in the object's frame of reference to $\boldsymbol{t}_{OC} = -r\boldsymbol{x}$. The rotation matrix that parameterises the orientation of the camera is then constructed so that the camera's optical axis intersects with the point $\boldsymbol{t}_{OC} + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim N(0, \sigma^2 \boldsymbol{I})$ using a method that is analogous to the OpenGL gluLookAt function [9]. In single image experiments, $\sigma = 1$ cm and in the trajectory-based experiment $\sigma = 0.2 d_{obj}$ where d_{obj} is an object's diameter. The algorithm then transforms the camera position in the object's frame to the object's position in the camera's frame and constructs the final pose of the camera.

Algorithm 2: Sampling object poses

```
Input: Object diameter $d_{obj}$ and maximum camera to
       object distance $d_{max}$
Output: Object pose in the camera's frame of
        reference $\boldsymbol{T}_{CO} = [\boldsymbol{R}_{CO} | \boldsymbol{T}_{CO}]$
Sample a scalar $r \sim \mathcal{U}[d_{obj}, d_{max}]$
Sample a point $\boldsymbol{x}$ uniformly distributed on the
  surface of a unit sphere
Set the camera position to $\boldsymbol{t}_{OC} = r\boldsymbol{x}$
Construct a rotation matrix $\boldsymbol{R}_{CO}$ such that the
  camera's optical axis intersects the point $\boldsymbol{t}_{OC} + \boldsymbol{\epsilon}$
  where $\boldsymbol{\epsilon} \sim N(0, \sigma^2 \boldsymbol{I})$
Set $\boldsymbol{T}_{CO} = [\boldsymbol{R}_{CO} | -\boldsymbol{R}_{CO}\boldsymbol{t}_{OC}]$
```

E. Sampling ICP initialisation poses

Given a ground truth object pose $\boldsymbol{T}_{CO}$, ICP initialisation was sampled using algorithm 3 that independently perturbs the position and orientation of an object. This algorithm begins by sampling two random vectors from the surface of a unit sphere. The first random vector, $\boldsymbol{v}$ and the desired magnitude of the rotation perturbation $\delta\theta$ are concatenated to form an axis-angle representation of a rotation perturbation. They are then converted to a matrix $\boldsymbol{R}_\delta \in SO(3)$. This rotation perturbation matrix is then applied on the right of the ground truth orientation of the object. Intuitively, this can be interpreted as rotating the ground truth orientation about a random axis $\boldsymbol{v}$ by an angle $\delta\theta$ in the object's frame

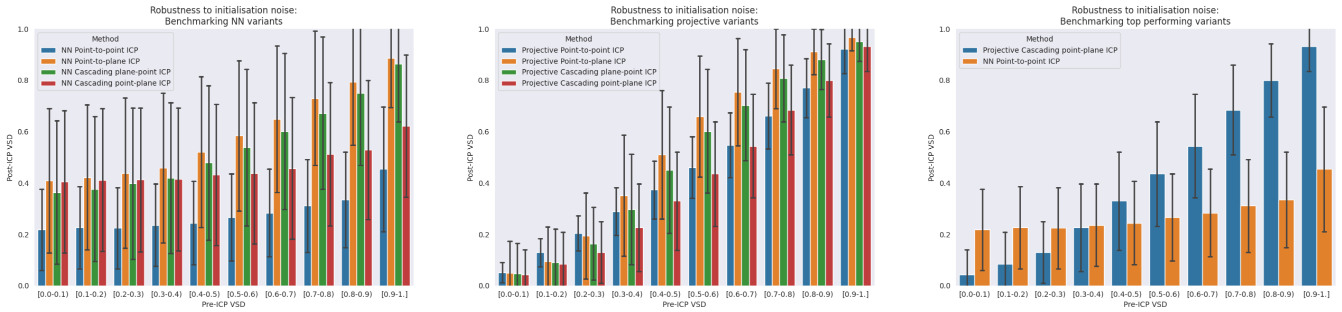

Fig. 1: Investigating the robustness of point-to-point, point-to-plane, Cascading point-plane and Cascading plane-point ICP with both NN and projective data association to noise in ICP initialisation. The leftmost graph benchmarks all ICP variants that use NN data association. The middle graph benchmarks all ICP variants that use projective data association. The rightmost graph benchmarks the performance of the top-performing NN variant against the performance of the top-performing projective variant.

of reference. The second random vector $\boldsymbol{\eta}$ is multiplied by the desired magnitude of the translation perturbation, and the resulting vector is added to the position of the object in the camera's frame of reference.

```
Algorithm 3: Sample initialisation pose
  Input: Ground truth pose of object in camera frame
         $\boldsymbol{T}_{CO} = [\boldsymbol{R}_{CO}|\boldsymbol{t}_{CO}]$ and magnitude of
         translation and rotation perturbation $\delta t$ and $\delta\theta$
  Output: ICP initialisation $\tilde{\boldsymbol{T}}^0_{CO}$ with the desired
          translational and rotational error
  Sample two random vectors $\boldsymbol{v}$ and $\boldsymbol{\eta}$ uniformly
    distributed on the surface of a unit sphere
  Construct a rotation matrix $\boldsymbol{R}_\delta$ from the axis-angle
    vector $(\delta\theta, \boldsymbol{v})$
  Let $\delta\boldsymbol{t} \leftarrow \delta t\boldsymbol{\eta}$
  $\tilde{\boldsymbol{T}}_{CO} = [\boldsymbol{R}_{CO}\boldsymbol{R}_\delta|\boldsymbol{t}_{CO} + \delta\boldsymbol{t}]$
```

F. Relationship between translation and rotation errors of a sampling-based pose estimator

We first estimated the relationship between the magnitudes of translation and rotation errors of a sampling-based pose estimator to generate realistic initialisations for our main experiments. This pose estimator sampled random orientations, concatenated them with the mean position of a point cloud, and refined each pose with projective Cascading ICP. It then returned the estimate with the lowest MVE. We have conducted an experiment in which we used this pose estimator to estimate the poses of each of the 10 test objects (see bottom row of figure 2 in our main paper) to find an empirical distribution of absolute translation and rotation errors and of the ratio of the rotation error per 1 mm of translation error. The absolute rotation error was defined as the angle from the axis-angle representation. From these distributions, it was found that on average, estimates from the sampling-based pose estimator had an angular error of $1.92°$ per 1 mm of translation error. It was also found that the mean translation error was equal to 6.2 mm while the mean rotation error was equal to $9.5°$.

III. Tuning the Dynamic Switching Threshold

This section describes the experimental procedure that we have adopted to tune the Dynamic Switching threshold used in the Hybrid ICP algorithm.

A. Experimental Procedure

From section III-A.2 of our main paper, recall that Dynamic Switching uses the MVE (see section IV-A.2 of our main paper for definition) to optimise the choice of the data association method. For this reason, we benchmark point-to-point, point-to-plane, Cascading point-plane and Cascading plane-point ICP with both NN and projective data association using the same experimental procedure as the one described in section VI-B.1 of our main paper (i.e. we investigate the robustness of each of the ICP variants to initialisation noise). To ensure that the found Dynamic Switching threshold generalises to novel objects, we benchmark these variants on a different set of 10 objects to the ones used in our main experiments (see section II-B).

B. Results

The results for this experiment are illustrated in figure 1. The leftmost graph in this figure benchmarks the performance of all NN ICP variants. As this graph illustrates, NN point-to-point ICP is the most robust variant to initialisation noise. We hypothesise that this is because (1) NN data association yields many incorrect correspondences when associating points sampled from thin object surfaces, and (2) the remaining variants all rely on point-to-plane ICP in one way or another which is prone to diverging when given incorrect correspondences when aligning point clouds with few geometric features. This result motivated using NN point-to-point ICP in Hybrid ICP.

The middle graph benchmarks all projective ICP variants. As this graph illustrates, Cascading point-plane is more robust to initialisation and object geometries than other projective ICP variants. This is because Cascading point-plane ICP refines ICP initialisation with point-to-point ICP prior to aligning two input point clouds with point-to-plane ICP. Combined with geometrically consistent correspondences obtained from projective data association, this minimises the probability of point-to-plane ICP diverging in the second stage of Cascading ICP.

The rightmost graph in this figure compares the performance of NN point-to-point ICP against that of projective Cascading point-plane ICP. The pre-ICP VSD error threshold at which point-to-point NN ICP begins to outperform projective cascading plane-point ICP can be used to set the Dynamic Switching threshold (in this case $\alpha = 0.4$). As shown in section VI-B of our main paper, this threshold enables Dynamic Switching to generalise to unseen objects.

References

[1] T. Barfoot *et al.*, “Associating uncertainty with three-dimensional poses for use in estimation problems,” *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, 2014.

[2] M. Brossard *et al.*, “A new approach to 3d icp covariance estimation,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 744–751, 2020.

[3] T. Hodaň *et al.*, “BOP toolkit,” 2020. [Online]. Available: https://github.com/thodan/bop_toolkit

[4] Q. Zhou *et al.*, “Open3D: A modern library for 3D data processing,” *arXiv:1801.09847*, 2018.

[5] R. Newcombe and othersw, “Kinectfusion: Real-time dense surface mapping and tracking,” in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136.

[6] A. Chang *et al.*, “ShapeNet: An Information-Rich 3D Model Repository,” *arXiv e-prints*, p. arXiv:1512.03012, Dec. 2015.

[7] B. Curless *et al.*, “A volumetric method for building complex models from range images,” in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’96. New York, NY, USA: Association for Computing Machinery, 1996, p. 303–312.

[8] W. Lorensen *et al.*, “Marching cubes: A high resolution 3D surface construction algorithm,” in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’87. New York, NY, USA: Association for Computing Machinery, 1987, p. 163–169.

[9] O. Wiki, “Opengl glulookat,” 2018. [Online]. Available: https://www.khronos.org/opengl/wiki/GluLookAt_code