A Lie Group Notations

In this section, we explain the notation used throughout this paper as well as the required preliminaries. Matrices are capitalized in bold, such as in $\mathbf{X}$, and vectors are in lower case bold type, such as in $\mathbf{x}$. Vectors are column-wise and $1:n$ means the integers from 1 to n. $\mathrm{vec}(x_1,\dots,x_n)$ denotes a vector such as $\mathbf{x}$ constructed by stacking x_i, $\forall\, i \in \{1:n\}$. An alphabet such as $\mathcal{X}$ denotes a set. The Euclidean norm is shown by $\|\cdot\|$. $\|\mathbf{e}\|^2_\Sigma \triangleq \mathbf{e}^\mathsf{T}\Sigma^{-1}\mathbf{e}$. The n-by-n identity matrix is denoted by $\mathbf{I}_n$. $\mathbf{0}_n$ denotes the vector of zeros with dimensions n.

Thorough details of the covered topics in this section are available in [1, 13, 30]. The general linear group of degree n, denoted by $\mathrm{GL}_n(\mathbb{R})$, is the set of all $n\times n$ nonsingular real matrices, where the group binary operation is the ordinary matrix multiplication. The 3D special orthogonal group, denoted by

$$\mathrm{SO}(3) = \{\mathbf{R}\in\mathrm{GL}_3(\mathbb{R}) | \mathbf{R}\mathbf{R}^\mathsf{T} = \mathbf{I}_3, \det\mathbf{R} = +1\},$$

is the rotation group on $\mathbb{R}^3$. The 3D special Euclidean group, denoted by

$$\mathrm{SE}(3) = \{\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}_3^\mathsf{T} & 1 \end{bmatrix} \in \mathrm{GL}_4(\mathbb{R}) | \mathbf{R}\in\mathrm{SO}(3), \mathbf{p}\in\mathbb{R}^3\},$$

is the group of rigid transformations on $\mathbb{R}^3$. The Lie algebra (tangent space at the identity together with Lie bracket) of SO(3), denoted by $\mathfrak{so}(3)$, is the set of 3×3 skew-symmetric matrices such that for any $\omega \triangleq \mathrm{vec}(\omega_1,\omega_2,\omega_3)\in\mathbb{R}^3$: $\omega^\wedge \triangleq \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$ and $(\omega^\wedge)^\vee = \omega$. The Lie algebra of SE(3), denoted by $\mathfrak{se}(3)$, can be identified by 4×4 matrices such that for any $\omega,\mathbf{v}\in\mathbb{R}^3$ and $\xi \triangleq \mathrm{vec}(\omega,\mathbf{v})\in\mathbb{R}^6$: $\xi^\wedge \triangleq \begin{bmatrix} \omega^\wedge & \mathbf{v} \\ \mathbf{0}_3^\mathsf{T} & 0 \end{bmatrix}$. The exponential map $\exp:\mathfrak{se}(3)\to\mathrm{SE}(3)$ can be used to map a member of $\mathfrak{se}(3)$ around a neighborhood of zero to a member of SE(3) around a neighborhood of the identity. The logarithm map is the inverse, i.e. $\log:\mathrm{SE}(3)\to\mathfrak{se}(3)$, and $\exp(\log(\mathbf{T})) = \mathbf{T}$. Now we can define the difference between a transformation $\mathbf{T}\in\mathrm{SE}(3)$ and its estimate with a small perturbation $\hat{\mathbf{T}}\in\mathrm{SE}(3)$ as [4, 13]:

$$\varepsilon^\wedge = \log(\hat{\mathbf{T}}\mathbf{T}^{-1})$$

where $\varepsilon^\wedge\in\mathfrak{se}(3)$. To define the norm of the error term, we exploit the fact that $\mathfrak{se}(3)$ is isomorphic to $\mathbb{R}^6$, i.e. $\varepsilon^\wedge \mapsto \varepsilon\in\mathbb{R}^6$ using the $\vee$ operator. Thus $\|\varepsilon\| = \|\log(\hat{\mathbf{T}}\mathbf{T}^{-1})^\vee\|$, and we define $\|\varepsilon\|^2_\Sigma \triangleq \varepsilon^\mathsf{T}\Sigma^{-1}\varepsilon$.

In optimization problem over a Lie group such as SE(3), the incremental term lives in the tangent space, and a retraction that maps it onto the Lie group is required [1]. For SE(3), the exponential map can serve as this retraction, and we solve the optimization problem by iteratively lifting (logarithm map) the cost function to the tangent space, solving the reparameterized problem, and then mapping the updated solution back to the original space using the retraction. For this work, we use the open source library Ceres Solver [2]. Using its local parametrizations, we can solve the nonlinear least squares problem by going to and from the tangent space of SE(3).

Table 3: Parameters used for each algorithm, similar values were chosen when possible, with the exception of IPDA which has a slightly different framework, for which we stayed close to the parameters in the authors implementation [17].

Parameters	Semantic ICP	GICP-SE(3)	GICP	IPDA
Convergence Threshold ε	1e−5	1e−5	1e−5	1e−3
Outer Max Iterations	50	50	50	50
Inner Max Iterations	200	200	200	100
Solver Backend	Ceres	Ceres	PCL	Ceres
Solver Algorithm	LM	LM	BFGS	LM
Jacobian	Analytical	Analytical	Analytical	Auto Diff
Parameter Representation	SE(3)	SE(3)	Euler Angles $\times \mathbb{R}^3$	SE(3)
Number of Threads	8	8	1	8
Distribution NN	20	20	20	NA
EM NN	4	NA	NA	4
NN Dist. Threshold	NA	NA	1.5 m	1.5 m
Cauchy Loss α	2.0	2.0	NA	NA

Table 4: Dilation CNN performance measure on the KITTI Odomentry Dataset

	Global Acc	Class Average Acc	mIoU	Inference Time (ms/image)
Dilation CNN	0.9738	0.9242	0.8482	214

B Optimization

This section will present the specifics of our evaluation, including parameters used by each algorithm and the hardware they were run on. Table 3 lists the parameters of each algorithm used. Notable difference includes the use of Ceres Solver's automatic differentiation by IPDA [17]. The experiments were run with version 1.13 of Ceres Solver, 3.3.4 of the Eigen matrix library, and version 1.8.1 of the Point Cloud Library. Timing results are presented on a computer with an Intel Core i7-3770 CPU, Nvidia Titan X (Pascal) GPU, and 32 GB of RAM.

C Dataset Processing

C.1 Kitti Visual Odometry Dataset

Disparity maps are created by the LIBELAS algorithm presented by Geiger et al. [19] using the rectified stereo images. For segmentation results, we used Dilation CNN [45] with a model trained on the KITTI dataset. Table 4 shows the statistics of the CNN on the dataset. We ran our evaluation on sequence 5 of the dataset. For the results presented, we calculated a trajectory with each algorithm by aligning every third point cloud. We then compared these relative transformations to the ground truth trajectories provided in the dataset.

To model $p(s_k|\mathcal{X}, i_k)$, as in (10), we fit a generalized Bernouli distribution to the same NN used to fit the Guassian residual distribution. That gives a distribution over CNN labels, to get the distribution over the true semantic class, we take the vector-matrix product of that

Table 5: DeepLab-ResNet performance measure on the SceneNet RGBD Dataset train_0

	Global Acc	Class Average Acc	mIoU	Inference Time (ms/image)
DeepLab-ResNet	0.8810	0.8453	0.7444	162

distribution with the normalized confusion matrix collected on the training data. That gives us another vector that is the generalized Bernouli distribution of true semantic class in that area.

C.2 SceneNet RGBD

Evaluation was performed on the validation portion of the dataset. The dataset provides ground-truth depth, for evaluation we added independent Gaussian noise to each depth measurement $n_{\texttt{Depth}} \sim \mathcal{N}(0,(0.04m)^2)$. Four trajectories were used (29, 223, 530, and 784).

DeepLab system re-purposes image classification networks for semantic segmentation by applying atrous convolution with upsampling filters, and yields significant improvement over its baselines. We chose DeepLab-ResNet, which is built on a re-purposed ResNet-101 [23], as the framework for semantic inference on SceneNet RGBD dataset. We initialized the model with weights pre-trained on MS-COCO dataset [25], and fine-tuned it on the first training set (train_0) of the SceneNet RGBD dataset which includes 300000 images. The semantic annotations are obtained by mapping instance labels given in SceneNet RGBD to NYUv2 13 class semantic labels [16].

The network was trained using the standard stochastic gradient descent algorithm and the “poly" learning rate policy with the base learning rate set to 0.00025 and power to 0.9. Momentum and weight decay are set to 0.9 and 0.0005 respectively. We used a mini-batch size of 10, and trained the network for a total of 150K iterations for 3 days on an Nvidia TITAN X (Pascal). The performance of the network is shown in Table 5. For this network we modeled $p(s_k|\mathcal{X}, i_k)$ similarly to how we did for sub-appendix C.1

D Kitti Visual Odometry Extended Results

Figure 5 shows the qualitative results of running Semantic ICP and GICP on a series of KITTI point clouds and then projecting them into a common reference frame. It shows that misalignment with GICP causes echos of objects, while Semantic ICP produces crisp point clouds.

E SceneNet RGBD Dataset Extended Results

Figure 6 shows error distance of the various methods in CDF and box plots. It shows a tighter grouping than was presented for the KITTI visual odometry dataset, but with Semantic ICP showing improvement over GICP and GICP-SE(3). Like the mean error metric, these plots are affected by the long tail of error values present in this data.

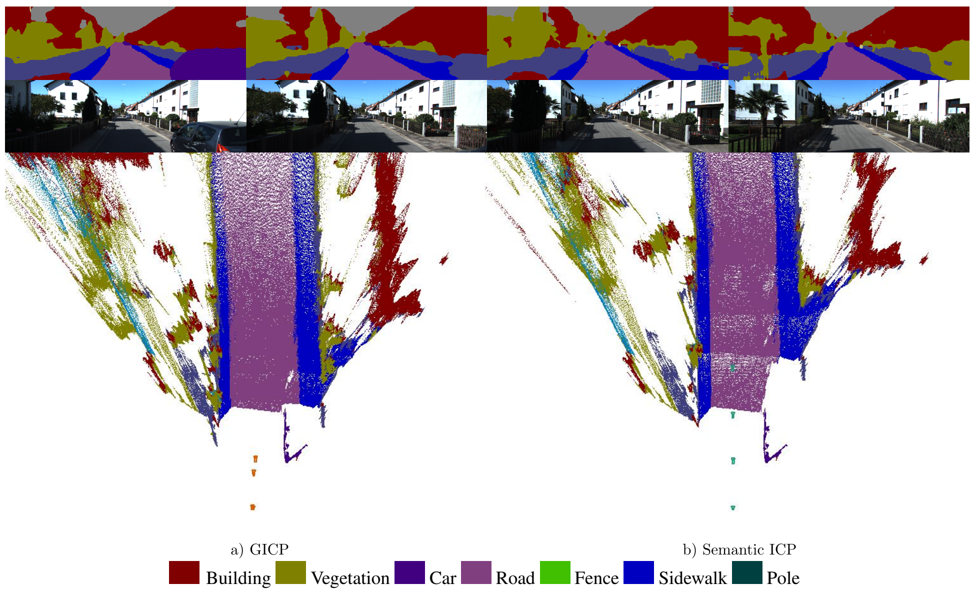

Figure 5: Sequential point clouds aligned using Semantic ICP on the right and GICP on the left. The top row shows the source image from the KITTI visual odometry dataset. The second row shows the inferred semantic labels produced using the Dilation CNN. The image on the left is the point clouds transformed by the estimated Semantic ICP transformations, with the camera positions marked in Cyan. The right are by the estimated GICP transformations with the camera positions marked in orange. The repeated object on the right side of the roadway are artifacts of poor alignment by GICP.

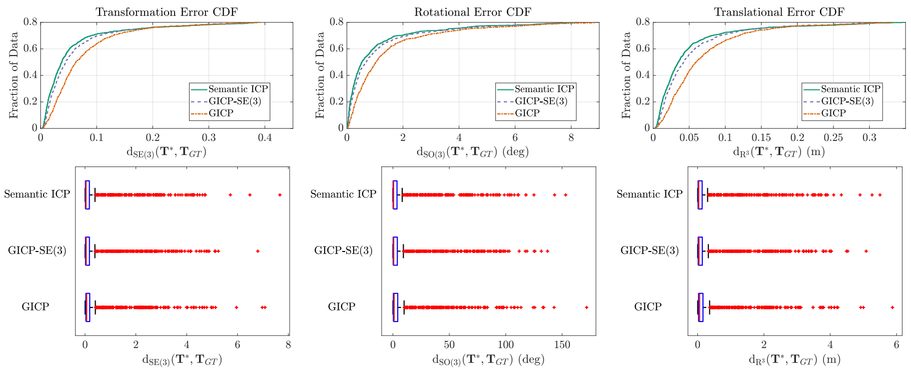

Figure 6: Box plots and cumulative distribution functions for the SceneNet RGBD dataset. There is a long tail on the errors for all methods for this dataset. The limited field of view and close-in objects led to high angle error. The CDFs are cut at 80% of the data to make the difference more clear but all data is visible in the box plots.