1. Algorithm Details

This section provides supplemental technical and implementation details in support of Section 4 of the paper.

1.1. Preprocessing

Extracting Features

As described in the main paper, for each input image \(I[k]\) we extract a set of features \(F[k]\). In our implementation we use 4 feature types:

1. **Plane** All parts of the image that are locally planar, i.e. can be represented as a position and a normal. We construct a planar surface feature for each pixel in coplanar clusters of size \(\geq 100\).
2. **Silhouette** We have found it useful to mark depth discontinuities with a separate feature type. These are represented as a point, normal and a direction along the silhouette. Normal and direction are estimated with PCA on neighboring silhouette pixels. The same approach is used to estimate direction for **Ridge** and **Valley** features.
3. **Ridge** If two large planes are intersecting and the angle between their normals is larger than \(\pi\) we mark the intersection as a ridge feature.
4. **Valley** - If two large planes are intersecting and the angle between their normals is smaller than \(\pi\) we mark them as a valley feature. Both valley and ridge features are represented as a position, normal (average of the normals of neighboring planes) and direction along the intersection.

We subsample the initial set of features using the Poisson Dart Algorithm, with a minimum spacing between the features equal to 0.05m. We first visit all the silhouette edges, then the ridges and valleys, followed by the planar features. This process allows us to create a uniformly sampled set of features for each image \(I[k]\), which improves the closest point constraint search times.

In our implementation we used a different weights to indicate the importance and our confidence for each feature type. Values used were \(\omega_{\text{plane}} = 1.0, \omega_{\text{silhouette}} = 0.6, \omega_{\text{ridge}} = 0.2, \omega_{\text{valley}} = 0.2\).

Creating Base Planar Proxies

To detect a set of planar proxies \(B\) in images \(I\) we have implemented our own agglomerative hierarchical clustering algorithm. We have found it to perform better that region growing or RANSAC algorithms on a noisy dataset like SUN3D. The low-level algorithmic details are:

- Apply a bilateral filter to reduce noise and quantization effects (\(\sigma_{xy} = 3\) px, \(\sigma_{\text{depth}} = 5\) cm).
- Mark pixels as boundaries if their depths differ from any of their neighbors by more than 10%.
- Compute connected components by partitioning the image on boundaries.
- Estimate normals for pixels using RANSAC on neighborhoods of radius \(r = 10\) cm within the same connected component.
- Compute sets of coplanar pixels using hierarchical clustering.
- Refine clusters with a RANSAC algorithm to reassign pixels to their largest compatible cluster.
- For each cluster insert proxy to \(B\). Assign the centroid, normal and radius for the proxy based on PCA of the associated pixels.

Each resulting planar proxy is represented by a centroid and a normal \(B[j] = \{p, \vec{n}\}\).

Aligning Adjacent Images

The frame-to-frame tracking that is used in our implementation is based on system proposed in Xiao et al.[10]. We begin by extracting matching SIFT features in consecutive RGB images. This process returns two sets of image locations \(P'_k\) and \(P'_{k+1}\). For each set of image locations we reject candidates that do not have valid depth value in the accompanying depth image, to create pruned out sets \(P_k\) and \(P_{k+1}\). Since all points in both sets now...
have valid depths, they can be back-projected into 3D using the intrinsics matrix $K$, to create $X_k$ and $X_{k+1}$. We use RANSAC to find a transformation that matches most of the back-projected points in order to create a pairwise transformation $L[k]$.

**Initializing Transformations** We then concatenate the transformations to create an initial set of poses for our optimization $T_0$.

$$(T_0[0] = I; T_0[j] = L[j]T_0[j - 1]; j \in [1:k])$$

**1.2. Fine-to-Coarse Refinement**

**Creating Co-planarity Constraints.** When clustering coplanar proxies $B$ we again use an agglomerative clustering algorithm that merges the transformed proxies based on their pairwise similarity. Similarity is expressed as a product of pairwise point-to-plane distance factors $f_{pp}$ and normal deviation factors $f_{nd}$. For pair of proxies $B_a, B_b$:

$$S_{ab} = f_{pp}(B_a, B_b)f_{pp}(B_b, B_a)f_{nd}(B_a, B_b)$$

Factors are expressed with respect to set maximum thresholds. In our implementation the maximum pairwise distance was set to $t_d = 30cm$ and the maximum normal deviation to $t_a = \frac{\pi}{8}$

$$f_{pp}(B_a, B_b) = \begin{cases} 1.0 - \frac{d(p_a, B_b)}{t_d} & \text{if } d(p_a, B_b) \geq t_d \\ 0.0 & \text{if } d(p_a, B_b) < t_d \end{cases}$$

$$f_{nd}(B_a, B_b) = \begin{cases} 1.0 - \frac{\angle(n_a, n_b)}{t_a} & \text{if } \angle(n_a, n_b) \geq t_a \\ 0.0 & \text{if } \angle(n_a, n_b) < t_a \end{cases}$$

Result of this procedure constitutes set of cluster proxies $P$.

**Creating Planar Relationship Constraints.** When deciding on the type of the relationship and its weight we use exponential function with $\sigma_\theta = 0.13$. This value will assign negligible weights to relationships that deviate more that $15^\circ$ from the desired angle. It is possible that this parameter should be adjusted iteratively, but we have not spent much time investigating or tweaking it.

**Creating Feature Correspondence Constraints.** For finding closest point correspondences we used the initial thresholds for distance and normal deviation were

$$\{d_{initial} = 0.5m, \alpha_{initial} = 30^\circ\}$$

and for final thresholds:

$$\{d_{final} = 0.2m, \alpha_{final} = 20^\circ\}$$

These values are adjusted based on the distance along-side trajectory between images whose features are currently considered. The initial values decrease as a square root with this distance, where the maximum is reached when the distance is equal to $0.5l_i$ and the minimum is when the distance is 0;

Although the error function presented in the paper is non-linear, we find that our overall algorithm converges more quickly when the variables are relaxed with a linear approximation once per iteration. To solve the system of linear equations we used the CSparse solver [4]. Though this approach is related to solving the non-linear function with a series of linear approximations (e.g., as is common in solvers like Ceres [1]), there is a significant difference — we update the structure of the error function between solving every linear approximation (by finding new structural and closest point constraints). This is beneficial, since solution to the updated function is generally closer to the final solution.

**1.3. Optimization**

For all our experiments we have run our algorithm using the same set of parameters:

- Starting window length $l_i = 3m$.
- Number of iterations $n_{iter} = 12$.
- Weights for energy function terms:
  - Initial: $w_C = 1500, w_L = 1000, w_H = 1500, w_G = 1500$
  - Final: $w_C = 1000, w_L = 1000, w_H = 1000, w_G = 1000$

Similarly to the feature correspondence thresholds, the weights are linearly interpolated between initial and final values throughout the optimization.

**Planar Structure Error.** The coplanar misalignment $E_{cp}$ of one cluster proxy $P_a$ to $P_b$ is computed by summing the squared distances from samples of points $q_s$ ($s \in [1, s_{max}]$). Each $q_s$ is either sampled from the boundary of a 1 meter radius disk around $p_a$, or is at the same location as $p_a$ (In our experiments $s_{max} = 5$):

$$E_{cp}^s(A, B) = \sum_{s=1}^{s_{max}} ((q_A - q_B) \cdot n_B)^2$$

$$E_{cp}(A, B) = E_{cp}^s(A, B) + E_{cp}^s(B, A)$$

**Local Alignment Error.** We compute the misalignment $E_l$ of one rigid transformation $T_j$ to another $T_k$ in the neighborhood of a point $c_j$ by summing the squared distances between points $p_s$ ($s \in [1, s_{max}]$)
sampled uniformly on a 1 meter radius sphere when they are transformed by $T_j$ versus $T_k$ ($s_{max} = 8$):

$$E_t(T_j, T_k) = \sum_{s=1}^{s_{max}} (T_j(p_s) - T_k(p_s))^2$$

These formulations provide error terms measured in squared distances between corresponding points (rather than differences of matrix elements, angles, viewpoints, etc.) and thus are more natural to combine with other error terms of the same form in our multi-objective optimization (as noted in [7]).

2. Ground Truth Correspondence Collection

To collect the set of correspondences required for our evaluation metric we have designed a user interface to assist in the collection process, as well as to evaluate the quality of correspondences. The interface can be seen in figure 1. The user is presented with view of image-pointcloud pairs. The user can freely scroll through the sequence to select corresponding pairs. Once a pair of images is marked the user can begin clicking points. Anytime a point is clicked the 3D view is also updated, which gives the user a sense where exactly a point lands in 3D space.

![Figure 1: Interface for collecting ground truth correspondences](image1)

Our interface also has an evaluation mode, which runs an iterative reconstruction algorithm with pairwise transformations and ground truth correspondences as constraints (Figure 2). We found this to be an easy way to validate that the clicked correspondences are correct, as an erroneous ground correspondence will lead to big errors in the resulting reconstruction and can be easily fixed by the user.

![Figure 2: Interface showing resulting reconstruction. On the side panel you can see colored button relating to pairs of images marked by the user.](image2)

3. Quantitative Results

Below we present the full set of results we used to produce a Table 1 in the paper. We also add results produced with two realtime tracking methods, Elastic Fusion [9] and Kintinuous [8]. Note that real-time methods are not expected to perform as well, due to the trade-offs one needs to make for a real-time system.

We would also like to note that Choi et al.[2] does not perform well on many sequences in our test set, as these sequences often are lacking strong geometrical features, and most of the tracking depends on RGB data. Since Robust Reconstruction [2] does not use the color channel, we do not expect it to be able to deal with such cases.

3.1. Robust Reconstruction

We have run the Robust Reconstruction[2] system with the default settings for SUN3D dataset, published by the authors in their downloadable binary package. The only change made was to modify the FragmentOptimizer.exe parameter from –slac to –rigid, which led to better results on SUN3D dataset. In running their code, we obtained slightly different results than those published in their paper. Though the difference is minor, we present both results in Table 2 to avoid confusion in other comparisons.

4. Qualitative Results

To demonstrate the robustness of our algorithm we show the results on the SceneNN [5], BundleFusion [3] and SUN3D [10] datasets.

**SceneNN** We show the resulting reconstructions of 6 SceneNN scenes (Figure 3), produced using Voxel-Hashing [6] with trajectories pre-computed by our algorithm. We also show the overhead pointcloud ren-
Table 1: Detailed quantitative results. RR abbreviates Robust Reconstruction by Choi et al.\[2\]. Errors are RMSE of distances of corresponding points given the trajectory produced by each method, in meters.

<table>
<thead>
<tr>
<th>Sequence Name</th>
<th>Ours</th>
<th>SUN3D</th>
<th>RR</th>
<th>Elastic Fusion</th>
<th>Kintinuous</th>
</tr>
</thead>
<tbody>
<tr>
<td>brown_bm_1</td>
<td>0.08345</td>
<td>0.25424</td>
<td>1.60400</td>
<td>1.90877</td>
<td>1.15671</td>
</tr>
<tr>
<td>brown_bm_4</td>
<td>0.10545</td>
<td>2.00690</td>
<td>4.12032</td>
<td>0.64936</td>
<td>1.78738</td>
</tr>
<tr>
<td>brown_cogsci_1</td>
<td>0.07161</td>
<td>0.89468</td>
<td>1.52869</td>
<td>0.75887</td>
<td>0.55985</td>
</tr>
<tr>
<td>brown_cs2</td>
<td>0.06346</td>
<td>0.21408</td>
<td>3.55556</td>
<td>0.89136</td>
<td>0.47414</td>
</tr>
<tr>
<td>brown_cs3</td>
<td>0.10796</td>
<td>1.90186</td>
<td>5.90101</td>
<td>2.90157</td>
<td>1.58114</td>
</tr>
<tr>
<td>hv_c11_2</td>
<td>0.06471</td>
<td>0.40341</td>
<td>0.27989</td>
<td>0.18390</td>
<td>0.15577</td>
</tr>
<tr>
<td>hv_c3_1</td>
<td>0.06541</td>
<td>0.30465</td>
<td>0.41692</td>
<td>0.30158</td>
<td>0.31309</td>
</tr>
<tr>
<td>hv_c5_1</td>
<td>0.07766</td>
<td>0.26991</td>
<td>0.11158</td>
<td>0.29152</td>
<td>0.28333</td>
</tr>
<tr>
<td>hv_c6_1</td>
<td>0.07524</td>
<td>0.62119</td>
<td>0.15189</td>
<td>0.27570</td>
<td>0.30313</td>
</tr>
<tr>
<td>hv_c8_3</td>
<td>0.08656</td>
<td>0.45715</td>
<td>0.24724</td>
<td>0.38132</td>
<td>0.28994</td>
</tr>
<tr>
<td>home_at_scan1_2013_jan_1</td>
<td>0.04063</td>
<td>0.21196</td>
<td>0.07570</td>
<td>1.18692</td>
<td>1.23930</td>
</tr>
<tr>
<td>home_bksh_oct_30_2012</td>
<td>0.05871</td>
<td>0.15002</td>
<td>1.23549</td>
<td>1.47723</td>
<td>0.58745</td>
</tr>
<tr>
<td>home_md_scan9</td>
<td>0.06063</td>
<td>0.16358</td>
<td>1.04740</td>
<td>1.29805</td>
<td>0.54559</td>
</tr>
<tr>
<td>nips_4</td>
<td>0.05109</td>
<td>0.15168</td>
<td>0.06181</td>
<td>0.45188</td>
<td>0.40953</td>
</tr>
<tr>
<td>scan1</td>
<td>0.06788</td>
<td>0.32143</td>
<td>1.19663</td>
<td>1.98147</td>
<td>1.46379</td>
</tr>
<tr>
<td>scan3</td>
<td>0.05042</td>
<td>0.07849</td>
<td>0.06207</td>
<td>0.13804</td>
<td>0.13694</td>
</tr>
<tr>
<td>maryland_hotel1</td>
<td>0.06140</td>
<td>0.30138</td>
<td>0.05156</td>
<td>0.65117</td>
<td>0.25950</td>
</tr>
<tr>
<td>maryland_hotel3</td>
<td>0.05794</td>
<td>0.20083</td>
<td>0.05260</td>
<td>0.15046</td>
<td>0.11797</td>
</tr>
<tr>
<td>d507_2</td>
<td>0.13874</td>
<td>0.32074</td>
<td>0.08354</td>
<td>0.57447</td>
<td>0.52683</td>
</tr>
<tr>
<td>ted_lab_2</td>
<td>0.04699</td>
<td>0.11556</td>
<td>0.05600</td>
<td>0.61538</td>
<td>0.59755</td>
</tr>
<tr>
<td>76-471b</td>
<td>0.04852</td>
<td>0.09020</td>
<td>0.04724</td>
<td>0.70408</td>
<td>0.68069</td>
</tr>
<tr>
<td>76-lstudyroom2</td>
<td>0.05347</td>
<td>0.17491</td>
<td>0.12469</td>
<td>0.55497</td>
<td>0.27545</td>
</tr>
<tr>
<td>dorm_next_sj</td>
<td>0.08661</td>
<td>0.21222</td>
<td>0.23403</td>
<td>0.19099</td>
<td>0.12923</td>
</tr>
<tr>
<td>lab_hj</td>
<td>0.09000</td>
<td>0.67366</td>
<td>0.10347</td>
<td>0.47529</td>
<td>0.16703</td>
</tr>
<tr>
<td>sc_athena</td>
<td>0.09680</td>
<td>0.13690</td>
<td>1.41592</td>
<td>1.40803</td>
<td>0.23490</td>
</tr>
</tbody>
</table>

Table 2: Result comparison between trajectories computed with Choi et al.\[2\](abbreviated as RR), and trajectories published at http://redwood-data.org/indoor/models.html. Errors are RMSE of distances of corresponding points given the trajectory produced by each method, in meters.

<table>
<thead>
<tr>
<th>Sequence Name</th>
<th>Ours</th>
<th>RR</th>
<th>RR(published)</th>
</tr>
</thead>
<tbody>
<tr>
<td>hv_c5_1</td>
<td>0.0689</td>
<td>0.1116</td>
<td>0.1938</td>
</tr>
<tr>
<td>hv_c6_1</td>
<td>0.0680</td>
<td>0.2669</td>
<td>0.2403</td>
</tr>
<tr>
<td>hv_c8_3</td>
<td>0.0731</td>
<td>0.2472</td>
<td>0.2007</td>
</tr>
<tr>
<td>maryland_hotel3</td>
<td>0.0606</td>
<td>0.0526</td>
<td>0.567</td>
</tr>
<tr>
<td>d507_2</td>
<td>0.1196</td>
<td>0.0835</td>
<td>0.2506</td>
</tr>
<tr>
<td>76-lstudyroom2</td>
<td>0.0567</td>
<td>0.1247</td>
<td>0.2497</td>
</tr>
<tr>
<td>dorm_next_sj</td>
<td>0.0905</td>
<td>0.2340</td>
<td>0.2583</td>
</tr>
<tr>
<td>lab_hj</td>
<td>0.0898</td>
<td>0.1035</td>
<td>0.1045</td>
</tr>
</tbody>
</table>

Table 1: Detailed quantitative results. RR abbreviates Robust Reconstruction by Choi et al.\[2\]. Errors are RMSE of distances of corresponding points given the trajectory produced by each method, in meters.

Table 2: Result comparison between trajectories computed with Choi et al.\[2\](abbreviated as RR), and trajectories published at http://redwood-data.org/indoor/models.html. Errors are RMSE of distances of corresponding points given the trajectory produced by each method, in meters.

We also show results of reconstructions of the 8 scenes published recently with BundleFusion\[3\](Figure 4). Our reconstructions match the quality of the results presented on the BundleFusion website\[1\], and are created with same parameters we used for our SUN3D evaluation. Again we show the qualitative results as there is neither ground truth information or code released for BundleFusion (as of writing of this document).

Lastly we also present visualizations of 374 scenes reconstructed from the SUN3D dataset. We found that our method produces good results in 350 cases (Figures 5,6,7,8,9,10,11,12). For 16 cases the resulting reconstructions have errors, mostly due to lost tracking by SUN3DSfM during the initialization.

\[1\]http://graphics.stanford.edu/projects/bundlefusion/
phase which leads to separate parts of the reconstruc-
tion that are aligned well, but that are not aligned with
each other (Figure 13). For 8 cases we have observed
some unexpected aberrations, most likely due to issues
with pairwise matching code producing initializations.
Figure 3: High quality renderings of SceneNN scenes. On the right hand side we show birds-eye view of the scene with regions marked, locating zoom-in views. Note that our alignment is able to produce reconstructions that both preserve the overall structure of the room, and low-scale geometric details.
Figure 4: Overhead renderings of BundleFusion scenes. Note how our results preserve underlying structure of the rooms, keeping the walls straight, while also producing high quality local alignments, comparable to those in [3].
Figure 5: Overhead pointcloud renderings of SUN3D scenes
Figure 6: Overhead pointcloud renderings of SUN3D scenes
Figure 7: Overhead pointcloud renderings of SUN3D scenes
Figure 8: Overhead pointcloud renderings of SUN3D scenes
Figure 9: Overhead pointcloud renderings of SUN3D scenes
Figure 10: Overhead pointcloud renderings of SUN3D scenes
Figure 11: Overhead pointcloud renderings of SUN3D scenes
Figure 12: Overhead pointcloud renderings of SUN3D scenes
Figure 13: Cases not reconstructing well due to lost tracking in initial alignment. If the transformation between frame $i$ and $i+1$ is highly incorrect, our reconstruction will not be able to recover and will produce erroneous results.
Figure 14: Overhead pointcloud renderings of SceneNN scenes
References


