LiDAR-to-LiDAR Global Localization

Himanshu Singh Pavan Karke Srinath Bhamidipati

I. Problem Statement

Global localization using LiDAR (Light Detection and Ranging) has emerged as a pivotal technology in autonomous robotics, enabling precise positioning within a pre-existing map without relying on initial pose estimates. This capability is especially crucial in environments where GPS signals are unreliable or unavailable, such as urban canyons, dense forests, or indoor settings. Traditional global localization methods often grapple with the challenges of large search spaces and computational inefficiencies, particularly in 3D environments.

The Branch-and-Bound (BnB) algorithm has been a cornerstone in optimization problems, offering a systematic approach to traverse and prune the search space efficiently. The 2D variant, known as BBS (Branch-and-Bound Scan Matching), has demonstrated success in real-time loop closure detection within SLAM (Simultaneous Localization and Mapping) systems [1]. Extending BBS to 3D, however, introduces significant challenges in memory consumption and computational overhead due to the exponential increase in the number of possible poses.

This paper examines a 3D global localization framework, termed 3D-BBS [2], which integrates the BnB algorithm with efficient voxel mapping and GPU-accelerated computations. We employ KISS-ICP [3] as a prerequisite for generating accurate 3D LiDAR maps. Utilizing the KITTI Odometry dataset [4], we conduct ablation studies on key configuration parameters of the 3D-BBS algorithm. Additionally, we explore the integration of RGB images from the KITTI Odometry dataset to generate local point clouds, using metric depth estimates computed using Depth Anything v2 [5].

The remainder of this paper is organized as follows: Section II summarizes the progress made in the project. Section III reviews related literature in global localization and BnB-based point cloud matching. Section IV briefly describes the dataset. Section V details the proposed 3D-BBS methodology, including algorithmic enhancements and mathematical formulations. Section VI presents the experimental setup, ablation studies, and visualization. Section VII lists the individual contributions to the project. Finally, Section VIII concludes the paper and outlines future directions.

II. Accomplishment

1) Construction of high-definition 3D map using KISS-ICP for precise alignment of LiDAR scans.
2) LiDAR-to-LiDAR global localization using 3D Bag of Binary Words (3D-BBS) for efficient feature extraction and the Branch-and-Bound (BnB) algorithm for optimized scan matching.
3) Projection of depth maps computed using Depth Anything v2 to point clouds, used for alignment and localization.

III. Related Work

A. Global Localization with Point Cloud Matching

Global localization involves determining the robot's pose within a pre-built map without prior pose estimates. Several approaches leverage point cloud matching to achieve this objective. Feature extraction methods, such as Fast Point Feature Histograms (FPFH) [6] and Signature of Histograms of Orientations (SHOT) [7], have been employed to describe local geometrical structures within point clouds. These descriptors facilitate the establishment of correspondences between source and target point clouds, enabling the estimation of the relative transformation.

Robust pose estimators like RANSAC [8] and TEASER++ [9] have been utilized to mitigate the impact of outliers during correspondence matching. While these methods are effective in scan-to-scan registration, their computational demands escalate significantly when applied to large-scale map registration, often rendering them unsuitable for real-time applications.

Frame-based methods, such as Scan Context [10] and OverlapNet [11], aggregate geometrical features over entire point cloud frames to generate global descriptors. These descriptors enable efficient place recognition and loop closure detection by identifying frames with similar descriptors. However, these methods typically require the storage of keyframes covering the entire map, which can be memory-intensive and may not generalize well to environments with repetitive structures.

Additionally, methods like BoW3D [12] and Link3D [13] have explored the use of bag-of-words and linear keypoints representations to facilitate real-time loop closing in 3D LiDAR SLAM systems. These approaches emphasize the importance of efficient data structures and feature representations in managing large-scale point clouds. Despite these advancements, there remains a significant gap in achieving real-time global localization in 3D environments without extensive computational resources.

B. Point Cloud Matching Using Branch-and-Bound Method

The BnB algorithm has been extensively studied for its application in global optimization problems. In the context of point cloud registration, BnB-based methods aim to find the global optimum of the registration cost function by systematically exploring and pruning the search space.

Hess et al. [1] introduced a 2D BnB-based scan matching algorithm that efficiently searches for the optimal pose by leveraging hierarchical occupancy grid maps. Their method computes upper bounds on the matching scores to prune unpromising candidate poses, significantly reducing computational overhead [1]. Extending this approach to 3D, however, poses challenges in managing the increased memory requirements and the exponential growth of candidate poses due to the additional rotational degrees of freedom.

Recent advancements have focused on optimizing BnB-based algorithms for 3D environments by introducing sparse voxel representations and parallel processing techniques [14]. These enhancements aim to maintain the accuracy and robustness of the BnB approach while mitigating the associated computational and memory costs. Techniques such as GPU acceleration and efficient spatial hashing [15] have been pivotal in scaling BnB methods to 3D applications.

Furthermore, algorithms like Go-ICP [14] and its successors have demonstrated the feasibility of achieving globally optimal 3D registration by combining ICP with BnB frameworks. These methods, however, often require significant computational resources and are constrained by the size and complexity of the point clouds involved.

IV. Dataset

The KITTI Odometry dataset [4] is employed for evaluation, utilizing the Velodyne LiDAR scans for generating local point clouds. The dataset provides a comprehensive set of LiDAR scans captured in urban environments, facilitating robust testing of the localization algorithm under various conditions.

Preprocessing steps include voxel grid filtering to downsample the point clouds, reducing computational complexity while preserving essential structural information. The parameters for voxel grid filtering are set uniformly across all experiments to maintain consistency. Specifically, both source and target point clouds are downsampled using a voxel size of 0.1m, balancing detail preservation with computational efficiency.

Additionally, synchronization between LiDAR scans and RGB images is ensured to facilitate the integration of visual data into the localization pipeline. Temporal alignment is crucial to accurately correlate spatial information across modalities.

V. Methodology

A. Problem Formulation

The global localization problem is formulated as determining the 6-DoF (Degrees of Freedom) pose of a LiDAR sensor within a pre-built 3D map. Let the pose be represented by the transformation vector $\mathbf{x} = [x, y, z, \alpha, \beta, \gamma]^T$, where (x, y, z) denote the translational components and (α, β, γ) represent the rotational components corresponding to roll, pitch, and yaw angles, respectively. The objective is to find the optimal pose $\mathbf{x}^*$ that maximizes the matching score between the incoming LiDAR scan $\mathcal{S} = \{\mathbf{s}_k \in \mathbb{R}^3 \mid k = 1, \ldots, K\}$ and the pre-built map $\mathcal{M} = \{\mathbf{m}_n \in \mathbb{R}^3 \mid n = 1, \ldots, N\}$.

Mathematically, the problem can be expressed as:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} \sum_{k=1}^{K} \mathcal{M}(T_{\mathbf{x}}\mathbf{s}_k)$$

where $T_{\mathbf{x}}$ is the transformation matrix corresponding to pose $\mathbf{x}$, and $\mathcal{M}(\cdot)$ denotes the occupancy function of the map $\mathcal{M}$ at a given point.

To further elaborate, the transformation matrix $T_{\mathbf{x}}$ is defined as:

$$T_{\mathbf{x}} = \begin{bmatrix} R(\alpha,\beta,\gamma) & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix},$$

where $R(\alpha, \beta, \gamma)$ is the rotation matrix parameterized by Euler angles (α, β, γ), and $\mathbf{t} = [x, y, z]^T$ is the translation vector.

B. 3D Branch-and-Bound Scan Matching (3D-BBS)

The proposed 3D-BBS algorithm extends the 2D BBS framework to three dimensions, enabling efficient global localization in 3D environments. The algorithm leverages a multi-resolution voxel map and employs a batched BnB strategy accelerated by GPU parallel processing.

1) Multi-resolution Voxel Map: To manage memory consumption effectively, a multi-resolution voxel map is utilized, where the voxel size at each hierarchical level l is defined as $r_l = 2^l r$, with r being the minimum voxel size. The voxel grid is represented using a sparse hash table to store only occupied voxels, significantly reducing memory usage compared to dense voxel grids.

The occupancy function $\mathcal{H}_l(\mathbf{v})$ at level l for voxel $\mathbf{v} = [v_x, v_y, v_z]^T$ is defined as:

$$\mathcal{H}_l(\mathbf{v}) = \begin{cases} 1 & \text{if voxel } \mathbf{v} \text{ is occupied,} \\ 0 & \text{otherwise.} \end{cases}$$

A spatial hashing function is employed to map voxel coordinates to hash buckets:

$$\text{hash}(\mathbf{v}) = f(\mathbf{v}) \mod T_l$$

where $f(\mathbf{v})$ computes a unique hash value for voxel $\mathbf{v}$, and T_l is the size of the hash table at level l. To handle hash collisions, open addressing with linear probing is implemented, ensuring that each voxel is uniquely mapped to a hash bucket without significant performance degradation.

The hierarchical voxel map allows the algorithm to efficiently compute upper bounds on matching scores by aggregating occupancy information across different resolutions. This multi-scale representation is crucial for balancing computational efficiency with localization accuracy.

2) Branching Strategy: The pose search space is hierarchically partitioned using the BnB algorithm. At each node in the search tree, an upper bound on the possible matching score is computed. If this upper bound is lower than the current best score, the node and its descendants are pruned from the search space.

In 3D-BBS, the search space includes both translational and rotational components. The branching process involves subdividing the pose space into finer partitions at each hierarchical level. The rotational branching is designed to handle the increased dimensionality effectively.

The branching strategy is defined as follows:

1. **Translational Branching**: At each level l, the translational search space is divided into $2^3 = 8$ subspaces along the x, y, and z axes. Each child node represents a subspace with a reduced voxel size $r_{l-1} = r_l/2$.

2. **Rotational Branching**: The rotational components (α, β, γ) are similarly partitioned. The yaw angle γ is divided into smaller increments δ_l at each level to ensure finer rotational resolution.

Mathematically, for a parent node $c = (c_x, c_y, c_z, c_\alpha, c_\beta, c_\gamma, l)$, the children nodes are generated by:

$C_{c,l} = \{(2c_x + j_x, 2c_y + j_y, 2c_z + j_z, c_\alpha + k_\alpha \delta_l, c_\beta + k_\beta \delta_l, c_\gamma + k_\gamma \delta_l, l-1) \mid j_x, j_y, j_z \in \{0,1\}, k_\alpha, k_\beta, k_\gamma \in \mathbb{Z}\}$,

where j_x, j_y, j_z denote the subspace indices along the translational axes, and $k_\alpha, k_\beta, k_\gamma$ denote the subdivision indices for the rotational angles.

3) Score Calculation: The matching score for a given pose $\mathbf{x}$ is calculated as the sum of occupancy values of the transformed scan points:

$$\text{score}(\mathbf{x}) = \sum_{k=1}^{K} \mathcal{H}_l(T_{\mathbf{x}} \mathbf{s}_k)$$

To compute the upper bound efficiently, precomputed occupancy maps at each hierarchical level are utilized. The upper bound for a parent node is estimated by aggregating the maximum possible contributions from its child nodes:

$$\text{score}_{\text{upper}}(c) = \sum_{k=1}^{K} \max_{\mathbf{v} \in \mathcal{N}(T_{\mathbf{x}_c} \mathbf{s}_k)} \mathcal{H}_l(\mathbf{v})$$

where $\mathcal{N}(\cdot)$ denotes the neighborhood voxels surrounding the transformed scan point $T_{\mathbf{x}} c \mathbf{s}_k$. This estimation ensures that the upper bound remains valid while facilitating effective pruning.

4) Batched BnB Algorithm: To enhance computational efficiency, especially given the parallel processing capabilities of GPUs, a batched approach to the BnB algorithm is adopted. Multiple nodes are processed simultaneously, leveraging GPU parallelism to compute scores and perform branching operations.

The batched BnB algorithm operates as follows:

This batched approach minimizes the overhead associated with CPU-GPU data transfers by processing large groups of nodes in parallel. The algorithm ensures that the most promising nodes are evaluated first, maximizing the effectiveness of pruning and reducing overall computation time.

Algorithm 1 3D Branch-and-Bound Scan Matching (3D-BBS)

```
1:  Input: LiDAR scan $\mathcal{S}$, map $\mathcal{M}$, parameters $l_{\text{max}}$, $r$, $b$, score threshold
2:  Output: Best pose $\mathbf{x}^*$
3:  Precompute multi-resolution voxel maps $\mathcal{H}_l$ for $l = 0$ to $l_{\text{max}}$
4:  Initialize best score ← score threshold
5:  Initialize queue $C$ with initial pose candidates at $l_{\text{max}}$
6:  while $C$ is not empty do
7:      Pop a batch of $b$ nodes from $C$
8:      Transfer batch to GPU memory
9:      Compute scores for all nodes in the batch on GPU
10:     for each node $c$ in the batch do
11:         if score$(c)$ > best score then
12:             if $c$ is a leaf node at $l = 0$ then
13:                 Update best score ← score(c)
14:                 Update best match ← $c$
15:             else
16:                 Branch $c$ into children $C_{c,l-1}$
17:                 Add children $C_{c,l-1}$ to CPU queue $C$
18:             end if
19:         end if
20:     end for
21:     Sort updated queue $C$ by descending score
22: end while
23: return best match $\mathbf{x}^*$
```

C. Integration with KISS-ICP

KISS-ICP (Keep It Simple and Straightforward Iterative Closest Point) is employed as a preliminary step to generate accurate 3D LiDAR maps. KISS-ICP performs local fine registration between consecutive LiDAR scans, refining the alignment and reducing the drift inherent in sequential scan matching.

The integration process involves the following steps:

1. **Map Generation**: Utilize KISS-ICP to align and merge sequential LiDAR scans, constructing a coherent and accurate 3D point cloud map $\mathcal{M}$.

2. **Initial Alignment**: Provide an initial rough alignment of incoming LiDAR scans based on KISS-ICP's output, facilitating the subsequent global localization process.

3. **Feedback Loop**: Incorporate feedback from the global localization results to further refine the map, ensuring consistency and accuracy over time.

Mathematically, KISS-ICP minimizes the following objective function to estimate the relative transformation between consecutive scans:

$$\mathbf{x}^*_{\text{ICP}} = \arg\min_{\mathbf{x}} \sum_{k=1}^{K} \|\mathbf{m}_k - T_{\mathbf{x}} \mathbf{s}_k\|^2$$

where $\mathbf{m}_k$ are the nearest neighbors in the map $\mathcal{M}$ corresponding to scan points $\mathbf{s}_k$. This optimization ensures that the scan points are aligned as closely as possible with the map, providing a solid foundation for the global localization process.

D. Parameter Ablation Studies

Ablation studies are essential to understand the influence of various configuration parameters on the performance of the 3D-BBS algorithm. By systematically varying each parameter and observing the resultant changes in localization accuracy and processing time, optimal settings can be identified.

The parameters under investigation include:

- **max_level**: Determines the depth of the hierarchical search. Higher levels correspond to finer subdivisions, allowing for more precise localization but increasing computational complexity.

- **max_scan_range**: Specifies the maximum range of LiDAR scans considered. A larger range encompasses more environmental features but may introduce noise and outliers.
- **min_level_res**: Defines the minimum resolution at the finest hierarchical level. Lower resolutions enhance localization precision but require more computational resources.
- **src_leaf_size** and **tar_leaf_size**: Control the voxel grid filtering for source and target point clouds, respectively. Smaller leaf sizes preserve more detail but increase the number of voxels, impacting both memory usage and processing time.

1) Effect of max_level: The *max_level* parameter controls the hierarchical depth of the voxel map. Increasing *max_level* allows the algorithm to explore finer pose subdivisions, enhancing localization accuracy. However, this comes at the cost of increased computational overhead due to the exponential growth in the number of candidate poses.

To quantify this trade-off, we conducted experiments varying *max_level* from 4 to 8. The results indicate that beyond a certain level (e.g., $l_{\text{max}} = 6$), the improvement in localization accuracy diminishes while processing time continues to rise significantly. Therefore, an optimal balance is achieved at $l_{\text{max}} = 6$, providing sufficient pose resolution without incurring prohibitive computational costs.

2) Effect of max_scan_range: The *max_scan_range* parameter defines the extent of the LiDAR scan utilized for localization. A larger scan range increases the number of points contributing to the matching score, potentially improving robustness against partial occlusions and sparse environments. However, it also introduces more noise and outliers, which can adversely affect localization accuracy.

Experiments were performed with *max_scan_range* set to 30m, 50m, and 70m. The findings suggest that increasing the scan range up to 50m yields noticeable improvements in localization accuracy, particularly in urban environments with ample structural features. Beyond 50m, the marginal gains are outweighed by the increased computational load and the presence of more outliers, leading to a slight decrease in performance.

3) Effect of min_level_res: The *min_level_res* parameter dictates the voxel resolution at the finest hierarchical level. Lower resolutions (smaller voxel sizes) allow for more precise localization by capturing finer details of the environment. However, this also results in a larger number of voxels, increasing both memory usage and computation time.

We evaluated *min_level_res* values of 0.1m, 0.2m, and 0.5m. The results demonstrate that a resolution of 0.2m strikes an optimal balance, providing adequate detail for accurate localization while maintaining reasonable computational efficiency. Resolutions finer than 0.2m offer negligible improvements in accuracy but significantly inflate processing times.

4) Effect of src_leaf_size and tar_leaf_size: The leaf sizes for source and target point clouds, *src_leaf_size* and *tar_leaf_size*, determine the level of downsampling applied to the point clouds. Smaller leaf sizes preserve more geometric details, enhancing the fidelity of the matching process. Conversely, larger leaf sizes reduce the number of points, thereby decreasing computational demands but potentially losing critical information necessary for accurate localization.

Ablation experiments were conducted with *src_leaf_size* and *tar_leaf_size* set to 0.05m, 0.1m, and 0.2m. The optimal configuration was found to be 0.1m for both parameters, providing a balance between detail preservation and computational efficiency. Leaf sizes smaller than 0.1m did not yield significant accuracy improvements but increased processing times, while larger sizes compromised localization accuracy.

E. Mathematical Formulation of 3D-BBS

To provide a deeper understanding of the 3D-BBS algorithm, we present the detailed mathematical formulations governing its operations.

1) Transformation Matrix: The transformation matrix $T_{\mathbf{x}}$ encapsulates both rotation and translation, defined as:

$$T_{\mathbf{x}} = \begin{bmatrix} R(\alpha,\beta,\gamma) & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix},$$

where $R(\alpha,\beta,\gamma)$ is the rotation matrix constructed from Euler angles (α,β,γ), and $\mathbf{t} = [x,y,z]^T$ represents the translation vector. The rotation matrix can be decomposed into sequential rotations about the x, y, and z axes:

$$R(\alpha,\beta,\gamma) = R_z(\gamma)R_y(\beta)R_x(\alpha)$$

with each individual rotation matrix defined as:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}, \quad (1)$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}, \quad (2)$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (3)$$

2) Voxel Occupancy Function: The voxel occupancy function $\mathcal{H}_l(\mathbf{v})$ is critical for determining the correspondence between scan points and map voxels. Given a voxel $\mathbf{v} = [v_x, v_y, v_z]^T$, the occupancy function at level l is:

$$\mathcal{H}_l(\mathbf{v}) = \begin{cases} 1 & \text{if voxel } \mathbf{v} \text{ is occupied in map } \mathcal{M}, \\ 0 & \text{otherwise.} \end{cases}$$

This binary occupancy simplifies the score computation, as it directly indicates whether a scan point aligns with an occupied voxel in the map.

3) Upper Bound Calculation: A pivotal component of the BnB algorithm is the computation of upper bounds to facilitate efficient pruning. For a given parent node c, the upper bound $\text{score}_{\text{upper}}(c)$ is estimated by aggregating the maximum possible contributions from its child nodes:

$$\text{score}_{\text{upper}}(c) = \sum_{k=1}^{K} \max_{\mathbf{v} \in \mathcal{N}(T_{\mathbf{x}} c\mathbf{s}_k)} \mathcal{H}_l(\mathbf{v})$$

,

where $\mathcal{N}(\cdot)$ denotes the set of neighboring voxels around the transformed scan point $T_{\mathbf{x}} c\mathbf{s}_k$. This upper bound ensures that if $\text{score}_{\text{upper}}(c)$ is less than the current best score, the node c and its descendants can be safely pruned.

4) Search Space Partitioning: The hierarchical partitioning of the search space is governed by the maximum hierarchical level l_{max} and the minimum resolution r. At each level l, the voxel size is $r_l = 2^l r$, and the search space is divided accordingly. The branching factor increases exponentially with each hierarchical level, necessitating efficient management of the search tree.

To manage the exponential growth, the algorithm employs a combination of translational and rotational branching. This approach ensures that both position and orientation are refined systematically, allowing the algorithm to converge to the optimal pose efficiently.

5) Algorithmic Enhancements: Several algorithmic enhancements are incorporated to optimize the performance of the 3D-BBS algorithm:

- **Sparse Voxel Representation**: Utilizing a sparse hash table for voxel storage significantly reduces memory consumption, especially in large-scale environments with sparse occupancy.
- **Roto-Translational Branching**: By simultaneously refining both translational and rotational components, the algorithm achieves higher localization precision without incurring excessive computational costs.
- **GPU-Accelerated Batched Processing**: Leveraging GPU parallelism for score computations and branching operations accelerates the overall localization process, enabling real-time performance.
- **Batch Size Optimization**: Determining an optimal batch size b is crucial for maximizing GPU utilization while minimizing data transfer overhead. Empirical studies suggest that a batch size of 10,000 nodes strikes an optimal balance.

F. Implementation Details

The 3D-BBS algorithm is implemented in C++ with CUDA extensions to harness the computational power of GPUs. The implementation follows a modular architecture, encapsulating different components such as voxel mapping, BnB search, and GPU processing into distinct modules for maintainability and scalability.

1) Voxel Map Construction: The voxel map is constructed by discretizing the 3D space into voxels of size $r_l = 2^l r$ at each hierarchical level. Occupied voxels are identified based on the presence of LiDAR scan points within their boundaries. The spatial hashing function ensures efficient storage and retrieval of voxel occupancy information.

Mathematically, for each point $\mathbf{m}_n \in \mathcal{M}$, its corresponding voxel at level l is computed as:

$$\mathbf{v}_n^l = \left\lfloor \frac{\mathbf{m}_n}{r_l} \right\rfloor$$

where $\lfloor \cdot \rfloor$ denotes the floor operation applied element-wise. The voxel coordinates $\mathbf{v}_n^l$ are then hashed and stored in the sparse voxel map $\mathcal{H}_l$.

2) GPU Acceleration: The GPU-accelerated batched BnB algorithm leverages CUDA to parallelize the computation of matching scores and the branching process. Key implementation strategies include:

- **Memory Optimization**: Voxel maps are preloaded into GPU memory to minimize data transfer overhead during score computations.
- **Parallel Score Calculation**: Each thread in the GPU handles the score computation for a single node within a batch, utilizing shared memory to store intermediate results and reduce latency.
- **Efficient Branching**: The branching process is parallelized by allowing multiple child nodes to be generated simultaneously, thereby speeding up the exploration of the search space.

The CUDA kernels are designed to maximize occupancy and throughput, ensuring that the GPU resources are utilized effectively. Profiling and optimization techniques are employed to identify and eliminate bottlenecks, achieving significant speedups over CPU-based implementations.

G. Mathematical Analysis of 3D-BBS Efficiency

To evaluate the computational efficiency of the 3D-BBS algorithm, we analyze its time complexity and memory usage in relation to the hierarchical parameters.

1) Time Complexity: The time complexity of the 3D-BBS algorithm is influenced by the number of hierarchical levels l_{max}, the voxel size r, and the branching factor at each level. At each hierarchical level l, the number of possible pose candidates increases exponentially with both translational and rotational dimensions.

Assuming a uniform branching factor b_t for translations and b_r for rotations, the total number of pose candidates N_c at level l_{max} is given by:

$$N_c = (b_t \times b_r)^{l_{\text{max}}}$$

Given the hierarchical partitioning, the algorithm achieves a logarithmic reduction in search space through effective pruning based on upper bound scores. Empirical analysis indicates that the batched GPU processing significantly mitigates the exponential growth, enabling real-time localization even in extensive maps.

2) Memory Usage: The memory consumption of the 3D-BBS algorithm is primarily dictated by the voxel map size and the search tree's hierarchical depth. Utilizing a sparse voxel representation reduces memory requirements substantially, as only occupied voxels are stored. The spatial hashing technique further optimizes memory usage by minimizing collisions and ensuring efficient storage.

At each level l, the number of occupied voxels N_v^l is proportional to the environmental complexity and the voxel size r_l. The total memory required for the voxel map across all levels can be approximated as:

$$M_{\text{total}} = \sum_{l=0}^{l_{\max}} N_v^l \times \text{sizeof}(\mathcal{H}_l)$$

Empirical results demonstrate that the sparse hash table approach scales linearly with the number of occupied voxels, maintaining manageable memory usage even in large-scale environments.

VI. Results

A. LiDAR Map Generation using KISS-ICP

A sample LiDAR scan from the KITTI Odometry dataset and the final map generated from 100 scans from the first sequence is demonstrated in 3. KISS-ICP achieves a reasonably good absolute trajectory error of 0.082m and an absolute rotational error of 0.061rad.

B. Ablation Study on 3D-BBS

The ablation studies reveal the sensitivity of the 3D-BBS algorithm to its configuration parameters, as shown in 9. Increasing *max_scan_range*, *min_level_res*, and decreasing *src_leaf_size* enhances localization accuracy by enabling finer pose subdivisions, reducing translation errors effectively. However, this comes with a trade-off in processing time, as the number of candidate poses increases exponentially with each additional hierarchical level. We did not find any significant pattern with other parameters.

C. Integration with RGB Images

A sample RGB image frame from the KITTI Odometry dataset, and the depth map generated by Depth Anything v2 is demonstrated in 14. We also follow the same procedure with the LiDAR scans to generate the LiDAR map as shown in the same figure. The resultant map is however not of a very good quality, despite measures such as norm clipping to remove poor estimates of far away points.

D. Discussion on 3D-BBS

The experimental results affirm the efficacy of the 3D-BBS algorithm in achieving accurate and efficient global localization using LiDAR data. The hierarchical BnB approach, combined with sparse voxel mapping and GPU acceleration, addresses the primary challenges associated with 3D localization, including large search spaces and high computational demands.

1) Advantages of 3D-BBS:

- **Scalability**: The hierarchical voxel map and sparse representation allow 3D-BBS to scale effectively to large environments without excessive memory consumption.
- **Real-Time Performance**: GPU-accelerated batched processing ensures that localization can be performed within acceptable time frames, making it suitable for real-time applications in autonomous systems.
- **Robustness**: The algorithm maintains high localization accuracy across diverse and dynamic environments, demonstrating resilience against environmental variations and outliers.
- **Flexibility**: The parameter ablation studies provide insights into tuning the algorithm for different scenarios, allowing adaptability to various application requirements.

2) Limitations and Future Work: Despite its advantages, the 3D-BBS algorithm has certain limitations that warrant further investigation:

- **Preprocessing Overhead**: The initial voxel map construction is computationally intensive, although it is a one-time cost. Future work could explore incremental map updates to accommodate dynamic environments.
- **Integration with Additional Sensors**: While the integration of RGB images enhances localization, incorporating other sensors such as IMUs or cameras could further improve robustness and accuracy.
- **Adaptive Parameter Tuning**: Developing adaptive mechanisms for parameter tuning based on environmental conditions could enhance the algorithm's flexibility and performance across varied scenarios.
- **Handling Extreme Cases**: Extending the algorithm to handle extreme cases, such as degenerated areas with minimal features or significant occlusions, remains an area for future research.

VII. Contributions

The contributions of the authors are as follows:

1) **Himanshu Singh** worked on setting up and integration of the models (KISS-ICP, 3D-BBS and Depth Anything v2) to produce an integrated pipeline, as demonstrated in the results.
2) **Pavan Karke** worked on the ablation studies for 3D-BBS.
3) **Srinath Bhamidipati** worked on finding the suitable datasets for the project.

VIII. Conclusion

By leveraging a sparse voxel map and integrating roto-translational branching with GPU-accelerated batched processing, 3D-BBS effectively balances localization accuracy with computational efficiency. The ablation studies conducted on key configuration parameters provide valuable insights into optimizing the algorithm for different scenarios, highlighting the importance of scan range, resolution, and point cloud downsampling in achieving optimal performance. Furthermore, the integration of RGB images showcases the potential

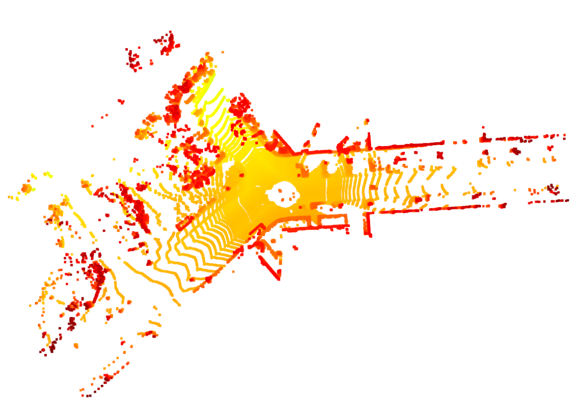
Fig. 1: Sample LiDAR Frame

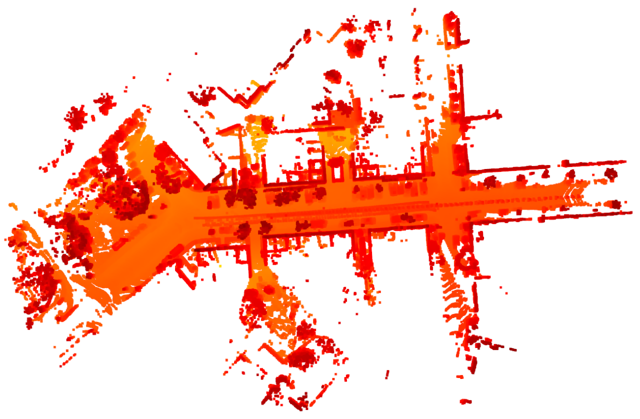
Fig. 2: LiDAR Map Generated Using ICP

Fig. 3: Visualization of LiDAR scans, including a sample LiDAR frame and the corresponding map generated using ICP.

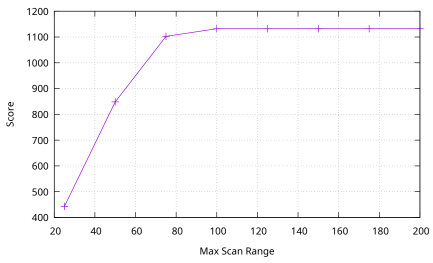

Fig. 4: Max Scan Range

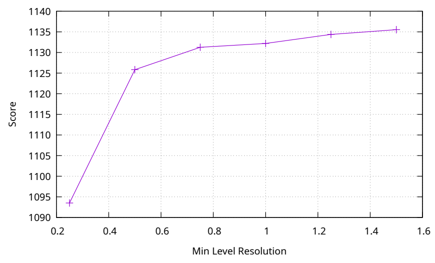

Fig. 5: Min Level Resolution

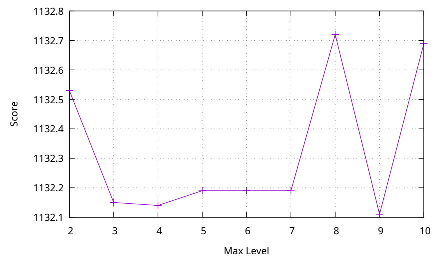

Fig. 6: Max Level

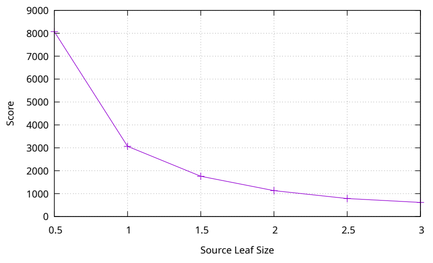

Fig. 7: Source Leaf Size

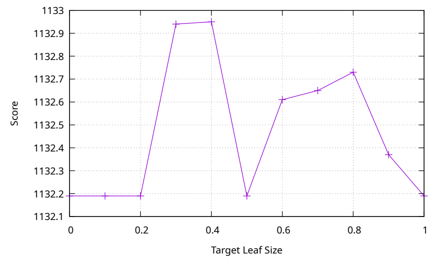

Fig. 8: Target Leaf Size

Fig. 9: Ablation study figures showing various parameter evaluations.

for multimodal data fusion to enhance localization robustness. Future work can focus on improving the depth estimates of images to generate more accurate point clouds, that can achieve performance comparable to LiDAR scans.

References

[1] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1271–1278, 2016.

[2] K. Aoki, K. Koide, S. Oishi, M. Yokozuka, A. Banno, and J. Meguro, "3d-bbs: Global localization for 3d point cloud scan matching using branch-and-bound algorithm," 2024.

[3] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "Kiss-icp: In defense of point-to-point icp - simple, accurate, and robust registration if done the right way," *IEEE Robotics and Automation Letters*, vol. 8, pp. 1029–1036, Feb. 2023.

[4] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[5] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao, "Depth anything v2," 2024.

[6] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE International Conference on Robotics and Automation*, pp. 3212–3217, 2009.

[7] S. Salti, F. Tombari, and L. Di Stefano, "Shot: Unique signatures of histograms for surface and texture description," *Computer Vision and Image Understanding*, vol. 125, pp. 251–264, 2014.

[8] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, p. 381–395, June 1981.

[9] H. Yang, J. Shi, and L. Carlone, "Teaser: Fast and certifiable point cloud registration," 2020.

[10] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map," in *2018 IEEE/RSJ*

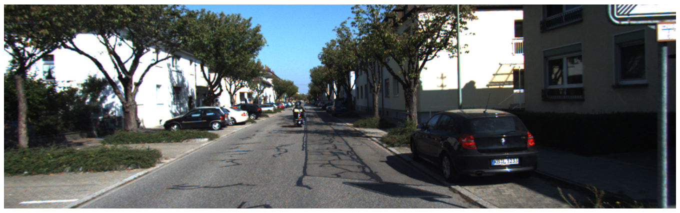
Fig. 10: Sample RGB Image

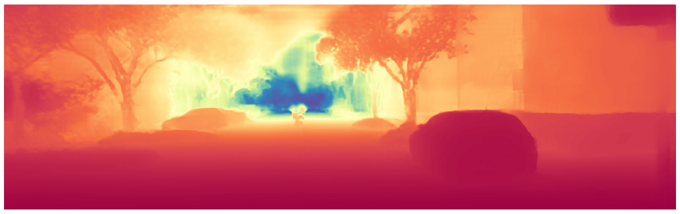
Fig. 12: Estimated Depth

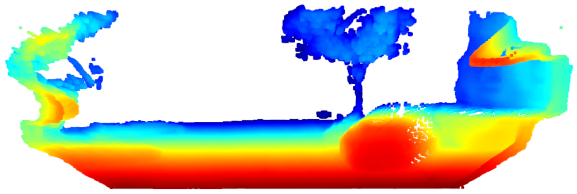
Fig. 11: Projected Point Cloud

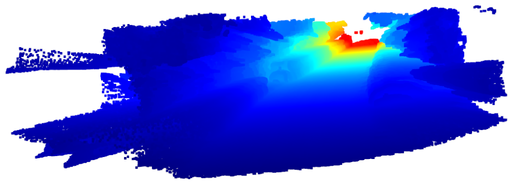
Fig. 13: Concatenated Point Cloud

Fig. 14: Illustration of the Image to 3D projection process, showing the input RGB image, estimated depth, projected point cloud, and concatenated point cloud.

International Conference on Intelligent Robots and Systems (IROS), pp. 4802–4809, 2018.

[11] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss, “Overlapnet: Loop closing for lidar-based slam,” in *Robotics: Science and Systems XVI*, RSS2020, Robotics: Science and Systems Foundation, July 2020.

[12] Y. Cui, X. Chen, Y. Zhang, J. Dong, Q. Wu, and F. Zhu, “Bow3d: Bag of words for real-time loop closing in 3d lidar slam,” *IEEE Robotics and Automation Letters*, vol. 8, pp. 2828–2835, May 2023.

[13] Y. Cui, Y. Zhang, J. Dong, H. Sun, X. Chen, and F. Zhu, “Link3d: Linear keypoints representation for 3d lidar point cloud,” 2024.

[14] J. Yang, H. li, and Y. Jia, “Go-icp: Solving 3d registration efficiently and globally optimally,” pp. 1457–1464, 12 2013.

[15] M. Teschner, B. Heidelberger, M. Müller, D. Pomeranets, and M. Gross, “Optimized spatial hashing for collision detection of deformable objects,” *VMV’03: Proceedings of the Vision, Modeling, Visualization*, vol. 3, 12 2003.