Online LiDAR-SLAM for Legged Robots with Robust Registration and Deep-Learned Loop Closure

Milad Ramezani, Georgi Tinchev, Egor Iuganov and Maurice Fallon

***Abstract*— In this paper, we present a 3D factor-graph LiDAR-SLAM system which incorporates a state-of-the-art deeply learned feature-based loop closure detector to enable a legged robot to localize and map in industrial environments. Point clouds are accumulated using an inertial-kinematic state estimator before being aligned using ICP registration. To close loops we use a loop proposal mechanism which matches individual *segments* between clouds. We trained a descriptor offline to match these segments. The efficiency of our method comes from carefully designing the network architecture to minimize the number of parameters such that this deep learning method can be deployed in real-time using only the CPU of a legged robot, a major contribution of this work. The set of odometry and loop closure factors are updated using pose graph optimization. Finally we present an efficient risk alignment prediction method which verifies the reliability of the registrations. Experimental results at an industrial facility demonstrated the robustness and flexibility of our system, including autonomous following paths derived from the SLAM map.**

I. Introduction

Robotic mapping and localization have been heavily studied over the last two decades and provide the perceptual basis for many different tasks such as motion planning, control and manipulation. A vast body of research has been carried out to allow a robot to determine where it is located in an unknown environment, to navigate and to accomplish tasks robustly online. Despite substantial progress, enabling an autonomous mobile robot to operate robustly for long time periods in complex environments, is still an active research area.

Visual SLAM has shown substantial progress [1], [2], with much work focusing on overcoming the challenge of changing lighting variations [3]. Instead, in this work we focus on LiDAR as our primary sensor modality. Laser measurements are actively illuminated and precisely sense the environment at long ranges which is attractive for accurate motion estimation and mapping. Hence, we focus on the LiDAR-SLAM specifically for legged robots.

The core odometry of our SLAM system is based on Iterative Closest Point (ICP), a well-known method for 3D shape registration [13], to align clouds from a 3D LiDAR sensor (Velodyne). Our approach builds upon our previous work of Autotuned ICP (AICP), originally proposed in [4], which analyzes the content of incoming clouds to robustify registration. Initialization of AICP is provided by the robot's state estimator [5], which estimates the robot pose by fusing the inertial measurements with the robot's kinematics and joint information in a recursive approach. Our LiDAR odometry has drift below 1.5% of distance traveled.

The authors are with the Oxford Robotics Institute (ORI), University of Oxford, UK. {milad, gtinchev, mfallon}@robots.ox.ac.uk, egor.iuganov@wadham.ox.ac.uk

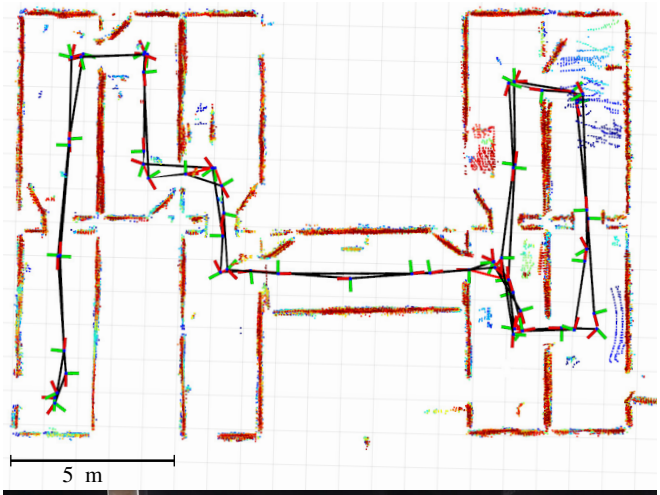


Fig. 1: Bird's-eye view of a map constructed by the ANYmal exploring an unlit, windowless, industrial facility with our proposed 3D SLAM system. The approach can rapidly detect and verify loop-closures (red lines) so as to construct an accurate map.

The contributions of our paper are summarized as follows:

1) An online 3D LiDAR-SLAM system for legged robots based on ICP registration.
2) A verification metric which quantifies the reliability of point cloud registration.
3) A demonstration of learned loop-closure detection designed to be deployable on a mobile CPU during run time.
4) A real-time demonstration of the algorithm on the ANYmal quadruped robot.

The remainder of this paper is structured as follows: Sec. II presents related works. Sec. III details different components of our SLAM system. Sec. IV presents evaluation studies before a conclusion and future works are drawn in Sec. V.

II. Related Works

This section provides a literature review of perception in walking robots and LiDAR-SLAM systems in general.

A. Perception systems on walking robots

Simultaneous Localization and Mapping (SLAM) is a key capability for walking robots and consequently their autonomy. Example systems include, the mono visual SLAM system of [6] which ran on the HRP-2 humanoid robot and is based on an Extended Kalman Filter (EKF) framework. In contrast, [7] leveraged a particle filter to estimate the posterior of their SLAM method, again on HRP-2. Oriolo *et al.* [8] demonstrated visual odometry on the Nano bipedal robot by tightly coupling visual information with the robot's kinematics and inertial sensing within an EKF. However, these approaches acquired only a sparse map of the environment on the fly, substantially limiting the robot's perception.

Works such as [9] and [10] leveraged the frame-to-model visual tracking of KinectFusion [11] or ElasticFusion [12], which use a coarse-to-fine ICP algorithm. Nevertheless, vision-based SLAM techniques struggle in varying illumination conditions. While robustness to illumination change has been explored, for example [14] and [3], it remains a major challenge to any visual navigation system.

Compared to bipedal robots, quadrupedal robots have better versatility and resilience when navigating challenging terrain. Thus, they are more suited to long-term tasks such as inspection of industrial sites. In an extreme case, a robot might need to jump over a terrain hurdle. Park *et al.*, in [15], used a Hokuyo laser range finder to detect 40 cm hurdles, which then allowed the MIT Cheetah-2's controller to jump over the obstacles, though the LiDAR measurements were not used for localization.

Nobili *et al.* [16] presented a state estimator for the Hydraulic Quadruped robot (HyQ) which was based on a modular inertial-driven EKF. The robot's base link velocity and position was propagated using IMU measurements with corrections from modules including leg odometry, visual odometry and LiDAR odometry. The system is prone to drift because the navigation system did not include a mechanism to detect loop-closures.

B. LiDAR-SLAM Systems

An effective LiDAR-based approach is LiDAR Odometry and Mapping (LOAM) [17]. LOAM extracts edge and surface point features in a LiDAR cloud by evaluating the roughness of the local surface. The features are reprojected to the start of the next scan based on a motion model, with point correspondences found within this next scan. Finally, the 3D motion is estimated recursively by minimizing the overall distances between point correspondences. LOAM achieves high accuracy with a low cost of computation.

Similar to LOAM, Deschaud in [18] developed a LiDAR odometry, IMLS-LiDAR, with similar accuracy by leveraging scan-to-model matching.

Shan *et al.* in [19] extended LOAM with Lightweight and Ground-Optimized LiDAR Odometry and Mapping (LeGO-LOAM) which added latitudinal and longitudinal parameters separately in a two-phase optimization. In addition, LeGO-LOAM detected loop-closures using an ICP algorithm to create a globally consistent pose graph using iSAM2 [20] (as part of the GTSAM library).

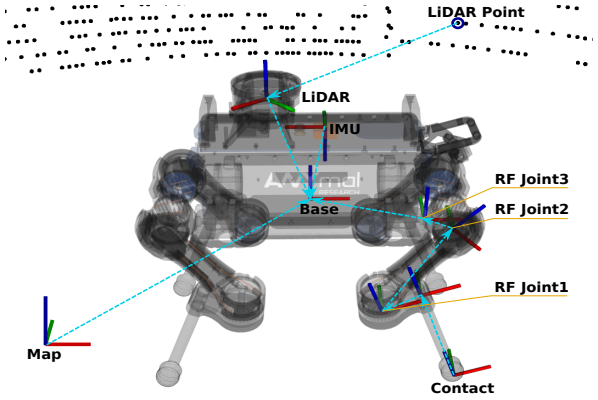


Fig. 2: Axis conventions of various frames used in the LiDAR-SLAM system and their relationship with respect to the base frame. As an example, we show only the Right Front (RF) leg.

Behley *et al.* in [21] proposed a surfel-based LiDAR-SLAM system by leveraging the projective data association between the current scan and a rendered model view from the surfel map.

Dubé *et al.* in [22] developed an online cooperative LiDAR-SLAM system. Using two and three robots, each equipped with 3D LiDAR, an incremental sparse pose-graph is populated by successive place recognition constraints. The place recognition constraints are identified utilizing the SegMatch algorithm [23], which represents LiDAR clouds as a set of compact yet discriminative features. The descriptors were used in a matching process to determine loop closures.

III. Approach

Our goal is to provide the ANYmal with a drift-free localization estimate over a very long mission, as well as to enable the robot to accurately map its surroundings. Fig. 2 and Fig. 3 elucidate the different components of our system from a hardware and a software perspective.

A. Kinematic-Inertial Odometry

We use the default state estimator for the robot to estimate incremental motion using proprioceptive sensing. This is the Two-State Implicit Filter (TSIF) [24]. The position of the contact feet in the inertial frame $\{\mathcal{I}\}$, obtained from forward kinematics, is treated as a temporal measurement to estimate the robot's pose in the fixed odometry frame, $\{\mathcal{O}\}$:

$$\mathbf{T}_{\mathcal{B}}^{\mathcal{O}} = \begin{bmatrix} \mathbf{R}_{\mathcal{B}}^{\mathcal{O}} & \mathbf{p}_{\mathcal{B}}^{\mathcal{O}} \\ 0 & 1 \end{bmatrix}, \tag{1}$$

where $\mathbf{T}_{\mathcal{B}}^{\mathcal{O}} \in SE(3)$ is the transformation from the base frame $\{\mathcal{B}\}$ to the odometry frame $\{\mathcal{O}\}$.

This estimate of the base frame drifts over time as it does not use any exteroceptive sensors and the position of the quadruped and its rotation around z-axis are not observable. In the following, we define our LiDAR-SLAM system for estimation of the robot's pose with respect to the map frame $\{\mathcal{M}\}$ which is our goal. It is worth noting that the TSIF framework estimates the covariance of the state [24] which we employ during our geometric loop-closure detection.

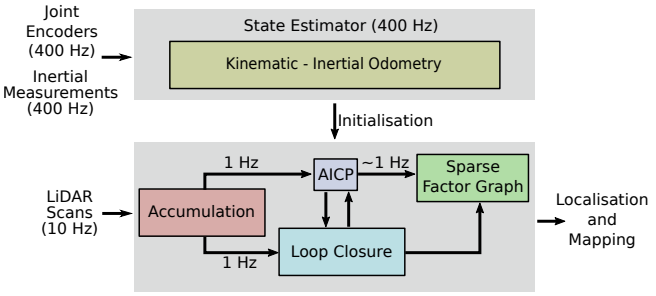


Fig. 3: Block diagram of the LiDAR-SLAM system.

B. LiDAR-SLAM System

Our LiDAR-SLAM system is a pose graph SLAM system built upon our ICP registration approach called Autotuned-ICP (AICP) [4]. AICP automatically adjusts the outlier filter of ICP by computing an overlap parameter, $\Omega \in [0, 1]$ since the assumption of a constant overlap, which is conventional in the standard outlier filters, violates the registration in real scenarios.

Given the initial estimated pose from the kinematic-inertial odometry, we obtain a **reference** cloud to which we align each consecutive **reading**[1] point cloud.

In this manner the successive reading clouds are precisely aligned with the reference clouds with greatly reduced drift. The robot's pose, corresponding to each reading cloud is obtained as follows:

$$\mathbf{T}_{\mathcal{B}}^{\mathcal{M}} = \Delta_{aicp}\mathbf{T}_{\mathcal{B}}^{\mathcal{O}}, \tag{2}$$

where $\mathbf{T}_{\mathcal{B}}^{\mathcal{M}}$ is the robot's pose in the map frame $\{\mathcal{M}\}$ and Δ_{aicp} is the alignment transformation calculated by AICP.

Calculating the corrected poses, corresponding to the point clouds, we compute the relative transformation between the successive reference clouds to create the odometry factors of the factor graph which we introduce in Eq. (3).

The odometry factor, $\phi_i(\mathcal{X}_{i-1}, \mathcal{X}_i)$, is defined as:

$$\phi_i(\mathcal{X}_{i-1}, \mathcal{X}_i) = (\mathbf{T}_{\mathcal{B}_{i-1}}^{\mathcal{M}^{-1}}\mathbf{T}_{\mathcal{B}_i}^{\mathcal{M}})^{-1}\widetilde{\mathbf{T}}_{\mathcal{B}_{i-1}}^{\mathcal{M}^{-1}}\widetilde{\mathbf{T}}_{\mathcal{B}_i}^{\mathcal{M}}, \tag{3}$$

where $\widetilde{\mathbf{T}}_{\mathcal{B}_{i-1}}^{\mathcal{M}}$ and $\widetilde{\mathbf{T}}_{\mathcal{B}_i}^{\mathcal{M}}$ are the AICP estimated poses of the robot for the node $\mathcal{X}_{i-1}$ and $\mathcal{X}_i$, respectively, and $\mathbf{T}_{\mathcal{B}_{i-1}}^{\mathcal{M}}$ and $\mathbf{T}_{\mathcal{B}_i}^{\mathcal{M}}$ are the noise-free transformations.

A prior factor, $\phi_0(\mathcal{X}_0)$, which is taken from the pose estimate of the kinematic-inertial odometry, is initially added to the factor graph to set an origin and a heading for the robot within the map frame $\{\mathcal{M}\}$.

To correct for odometric drift, loop-closure factors are added to the factor graph once the robot revisits an area of the environment. We implemented two approaches for proposing loop-closures: a) geometric proposal based on the distance between the current pose and poses already in the factor graph which is useful for smaller environments, and b) a learning approach for global loop-closure proposal, detailed in Sec III-C.1, which scales to large environments. Each proposal provides an initial guess, which is refined with ICP.

[1]Borrowing the notation from [25], the **reference** and **reading** clouds correspond to the robot's poses at the start and end of each edge in the factor graph and AICP registers the latter to the former.

Each individual loop closure becomes a factor and is added to the factor graph. The loop-closure factor, in this work, is a factor whose end is the current pose of the robot and whose start is one of the reference clouds, stored in the history of the robot's excursion. The nominated reference cloud must meet two criteria of nearest neighbourhood and sufficient overlap with the current reference cloud. An accepted loop-closure factor, $\phi_j(\mathcal{X}_M, \mathcal{X}_N)$, is defined as:

$$\phi_j(\mathcal{X}_M, \mathcal{X}_N) = (\mathbf{T}_{\mathcal{B}_M}^{\mathcal{M}^{-1}}\mathbf{T}_{\mathcal{B}_N}^{\mathcal{M}})^{-1}(\Delta_{j,aicp}\widetilde{\mathbf{T}}_{\mathcal{B}_M}^{\mathcal{M}})^{-1}\widetilde{\mathbf{T}}_{\mathcal{B}_N}^{\mathcal{M}}, \tag{4}$$

where $\widetilde{\mathbf{T}}_{\mathcal{B}_M}^{\mathcal{M}}$ and $\widetilde{\mathbf{T}}_{\mathcal{B}_N}^{\mathcal{M}}$ are the robot's poses in the map frame $\{\mathcal{M}\}$ corrected by AICP with respect to the reference cloud $(M-1)$ and the reference cloud $(N-1)$, respectively. The $\Delta_{j,aicp}$ is the AICP correction between the current reference cloud M and the nominated reference cloud N.

Once all the factors, including odometry and loop-closure factors, have been added to the factor graph, we optimize the graph so that we find the Maximum A Posteriori (MAP) estimate for the robot poses corresponding to the reference clouds. To carry out this inference over the variables $\mathcal{X}_i$, where i is the number of the robot's pose in the factor graph, the product of all the factors, must be maximized:

$$\mathcal{X}^{MAP} = \underset{\mathcal{X}}{\operatorname{argmax}} \prod_i \phi_i(\mathcal{X}_{i-1}, \mathcal{X}_i) \prod_j \phi_j(\mathcal{X}_M, \mathcal{X}_N). \tag{5}$$

Assuming that factors follow a Gaussian distribution and all measurements are only corrupted with white noise, i.e. noise with normal distribution and zero mean, the optimization problem in Eq. (5) is equivalent to minimizing a sum of nonlinear least squares:

$$\begin{aligned}\mathcal{X}^{MAP} = \underset{\mathcal{X}}{\operatorname{argmin}} &\sum_i ||y_i(\mathcal{X}_i, \mathcal{X}_{i-1}) - m_i||^2_{\mathbf{\Sigma}_i} \\ &+ \sum_j ||y_j(\mathcal{X}_M, \mathcal{X}_N) - m_j||^2_{\mathbf{\Sigma}_j},\end{aligned} \tag{6}$$

where m, y and $\mathbf{\Sigma}$ denote the measurements, their mathematical model and the covariance matrices, respectively.

As noted, the MAP estimate is only reliable when the residuals in Eq. (6) follow the normal distribution. However, ICP is susceptible to failure in the absence of geometric features, e.g. in corridors or door entries, which can have a detrimental effect when optimizing the pose graph. In Sec. III-D, we propose a fast verification technique for point cloud registration to detect possible failure of the AICP registration.

C. Loop Proposal Methods

This section introduces our learned loop-closure proposal and geometric loop-closure detection.

1) Deeply-learned Loop Closure Proposal: We use the method of Tinchev *et al.* [26], which is based on matching individual segments in pairs of point clouds using a deeply-learned feature descriptor. Its specific design uses a shallow network such that it does not require a GPU during runtime inference on the robot. We present a summary of the method called Efficient Segment Matching (ESM), but refer the reader to [26].

Algorithm 1: Improved Risk Alignment Prediction.

input: point clouds $\mathcal{C}_S$, $\mathcal{C}_T$; estimated poses $\mathcal{X}_S$, $\mathcal{X}_T$
output: alignment risk $\rho = f(\Omega, \alpha)$
begin
  Segment $\mathcal{C}_S$ and $\mathcal{C}_T$ into a set of planes: P_{Si} and P_{Tj},
  Compute centroid of each plane: $\boldsymbol{K}_{Si}$, $\boldsymbol{K}_{Tj}$,
  Transform query keypoints $\boldsymbol{K}_{Tj}$ into the space of $\mathcal{C}_S$,
  Search the nearest neighbour of each query plane P_{Tj},
  for *plane* P_T **do**
    Find match P_S amongst candidates in the k-d tree,
    Compute the matching score Ω_p,
    if Ω_p *is max* **then**
      Determine the normal of plane P_T,
      Push back the normal into the matrix $\boldsymbol{N}$,
    end
  end
  Compute alignability $\alpha = \lambda_{max} \,/\, \lambda_{min}$;
  Learn $\rho = f(\Omega, \alpha)$;
  Return ρ;
end

First, a neural network is trained offline using individual LiDAR observations (segments). By leveraging odometry in the process, we can match segment instances without manual intervention. The input to the network is a batch of triplets - anchor, positive and negative segments. The anchor and positive samples are the same object from two successive Velodyne scans, while the negative segment is a segment chosen $\approx 20\,\text{m}$ apart. The method then performs a series of X-conv operators directly on raw point cloud data, based on PointCNN [27], followed by three fully connected layers, where the last layer is used as the descriptor for the segments.

During our trials, when the SLAM system receives a new reference cloud, it is preprocessed and then segmented into a collection of point cloud clusters. For each segment in the reference cloud, a descriptor vector is computed with an efficient TensorFlow C++ implementation by performing a forward pass using the weights from the already trained model. This allows a batch of segments to be preprocessed simultaneously with zero-meaning and normalized variance and then forward passed through the trained model. We use a three dimensional tensor as input to the network - the length is the number of segments in the current point cloud, the width represents a fixed-length down-sampled vector of all the points in an individual segment, and the height contains the x, y and z values. Due to the efficiency of the method, we need not split the tensor into mini-batches, allowing us to process the full reference cloud in a single forward pass.

Once the descriptors for the reference cloud are computed, they are compared to the map of previous reference clouds. ESM uses an l_2 distance in feature space to detect matching segments and a robust estimator to retrieve a 6DoF pose. This produces a transformation of the current reference cloud with respect to the previous reference clouds. The transformation is then used in AICP to add a loop-closure as a constraint to the graph-based optimization. Finally, ESM's map representation is updated, when the optimization concludes.

2) Geometric Loop-Closure Detection: To geometrically detect loop-closures, we use the covariance of the legged state estimator (TSIF) to define a dynamic search window around the current pose of the robot. Then the previous robot's poses, which reside within the search window, are examined based on two criteria: nearest neighbourhood and verification of cloud registration (described in Sec. III-D). Finally, the geometric loop closure is computed between the current cloud and the cloud corresponding to the nominated pose using AICP.

D. Fast Verification of Point Cloud Registration

This section details a verification approach for ICP point cloud registration to determine if two point clouds can be safely registered. We improve upon our previously proposed alignability metric in [28] with a much faster method.

Method from [28]: First, the point clouds are segmented into a set of planar surfaces. Second, a matrix $\boldsymbol{N} \in \mathbb{R}^{M\times 3}$ is computed, where each row corresponds to the normal of the planes ordered by overlap. M is the number of matching planes in the overlap region between the two clouds. Finally, the alignability metric, α is defined as the ratio between the smallest and largest eigenvalues of $\boldsymbol{N}$.

The matching score, Ω_p, is computed as the overlap between two planes, P_{Tj} and P_{Si} where $i \in N_S$ and $j \in N_T$. N_S and N_T are the number of planes in the input clouds. In addition, in order to find the highest possible overlap, the algorithm iterates over all possible planes from two point clouds. This results in overall complexity $O(N_S N_T (N_{P_S} N_{P_T}))$, where N_{P_S} and N_{P_T} are the average number of points in planes of the two clouds.

Proposed Improvement: To reduce the pointwise computation, we first compute the centroids of each plane $\boldsymbol{K}_{Si}$ and $\boldsymbol{K}_{Tj}$ and align them from point cloud $\mathcal{C}_T$ to $\mathcal{C}_S$ given the computed transformation. We then store the centroids in a k-d tree and for each query plane $P_T \in \mathcal{C}_T$ we find the K nearest neighbours. We compute the overlap for the K nearest neighbours, and use the one with the highest overlap. This results in $O(N_T K (N_{P_S} N_{P_T}))$, where $K << N_s$. In practice, we found that $K = 1$ is sufficient for our experiments. Furthermore, we only store the centroids in a k-d tree, reducing the space complexity.

We discuss the performance of this algorithm, as well as its computational complexity in the experiment section of this paper. Pseudo code of the algorithm is available in Algorithm 1.

IV. Experimental Evaluation

We employ a state-of-the-art quadruped, ANYmal (version B) [29], as our experimental platform. Fig. 4 shows a view of the ANYmal robot. The robot weighs about 33 kg without

Sensor	Model	Frequency	Specifications
IMU	Xsens MTi-100	400	Bias Repeatability: $< 0.5°/s$; $5\ mg$ Bias Stability: $10°/h$; $40\ \mu g$
LiDAR Unit	Velodyne VLP-16	10	Resolution in Azimuth: $< 0.4°$ Resolution in Zenith: $2.0°$ Range $< 100\ m$ Accuracy: $\pm\ 3\ cm$
Encoder	ANYdrive	400	Resolution $< 0.025°$
Torque	ANYdrive	400	Resolution $< 0.1\ Nm$

TABLE I: Specifications of the sensors installed on the ANYmal.

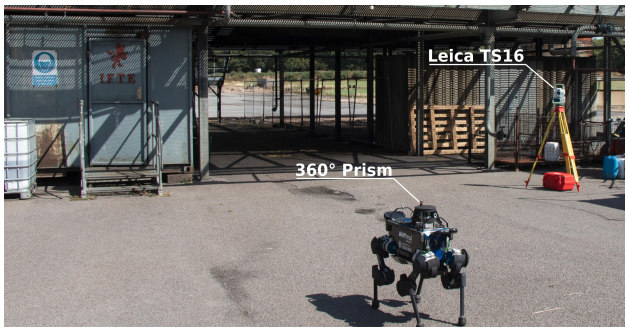


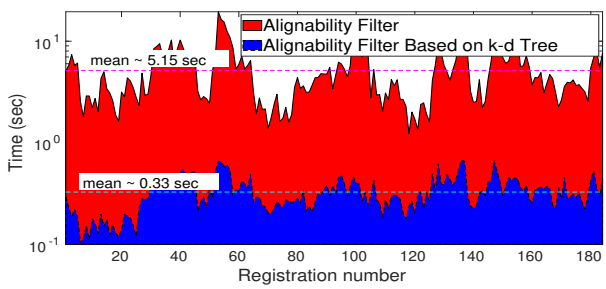
Fig. 4: Our experiments were carried out with the ANYmal, in the *Oil Rig* training site. To determine ground truth robot poses, we used a Leica TS16 laser tracker to track a 360° prism on the robot.

any external perception modules and can carry a maximum payload of 10 kg and trot at a maximum speed of 1.0 m/s. As shown in Fig. 2, each leg contains 3 actuated joints which altogether gives 18 degrees of freedom, 12 actuated joints and the 6 DoF robot base, which the robot uses to dynamically navigate challenging terrain. Tab. I summarizes the specifications of the robot's sensors used in this paper. The proposed LiDAR-SLAM system is evaluated using the datasets collected by our ANYmal quadruped robot.

We first analyze the verification method. Second, we investigate the learned loop-closure detection in terms of speed and reliability (Sec. IV-B). We then demonstrate the performance or our SLAM system on two large-scale experiments, one indoor and one outdoor (Sec. IV-C) including an online demonstration where the map is used for route following (Sec. IV-D). A demonstration video can be found at: `https://ori.ox.ac.uk/lidar-slam`.

A. Verification Performance

We focus on the alignability metric α of the alignment risk prediction since it is our primary contribution. We computed α between consecutive point clouds of our outdoor experiment, which we discuss later in Sec. IV-C. As seen in Fig. 6 (Left), the alignability filter based on a k-d tree is substantially faster than the original filter. As noted in Sec. III-D, our approach is less dependent on the point cloud size, due to only using the plane centroids. Whereas, the original alignability filter fully depends on all the points of the segments, resulting in higher computation time. Having tested our alignability filter for the datasets taken from Velodyne VLP-16, the average computation time is less than 0.5 seconds (almost 15x improvement) which is suitable for real-time operation.



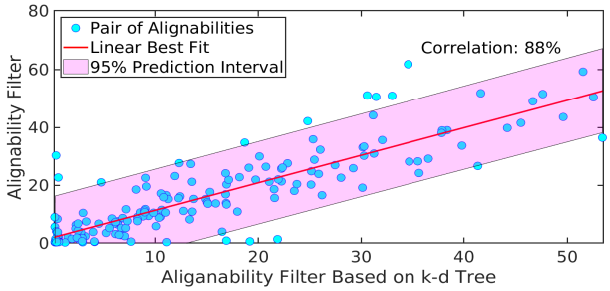


Fig. 6: Comparison of our proposed verification method with the original in terms of computation time (left) and performance (right).

Fig. 5: Demonstration of the learning-based loop closure in the outdoor experiment. See Sec. IV-B for more details.

Fig. 6 (Right) shows that our approach highly correlates with the result from the original approach. Finally, we refer the reader to the original work that provided a thorough comparison of alignment risk against Inverse Condition Number (ICN) [30] and Degeneracy parameter [31] in ill-conditioned scenarios. The former mathematically assessed the condition of the optimization, whereas the latter determines the degenerate dimension of the optimization.

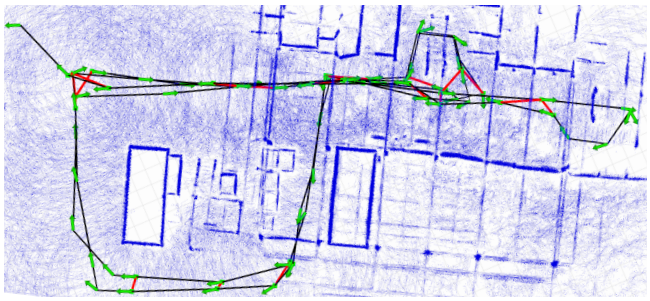
B. Evaluation of Learned Loop Closure Detector

In the next experiment we explore the performance of the different loop closure methods using a dataset collected outdoors at the Fire Service College, Moreton-in-Marsh, UK.

1) Robustness to viewpoint variation: Fig. 5 shows a preview of our SLAM system with the learned loop closure method. We selected just two point clouds to create the map, with their positions indicated by the blue arrows in Fig. 5.

The robot executed two runs in the environment, which comprised 249 point clouds. We deliberately chose to traverse an offset path (red) the second time so as to determine how robust our algorithm is to translation and viewpoint variation. In total 50 loop closures were detected (green arrows) around the two map point clouds. Interestingly, the approach not only detected loop closures from both trajectories, with translational offsets up to 6.5 m, but also with orientation variation up to 180°- something not achievable by standard visual localization - a primary motivation for using LiDAR. Across the 50 loop closures an average of 5.24 segments were recognised per point cloud. The computed transformation had an Root Mean Square Error (RMSE) of 0.08 ± 0.02 m from the ground truth alignment. This was achieved in approximately 486 ms per query point cloud.

2) Computation Time: Fig. 8 shows a graph of the computation time. The computation time for the geometric loop closure method depends on the number of traversals



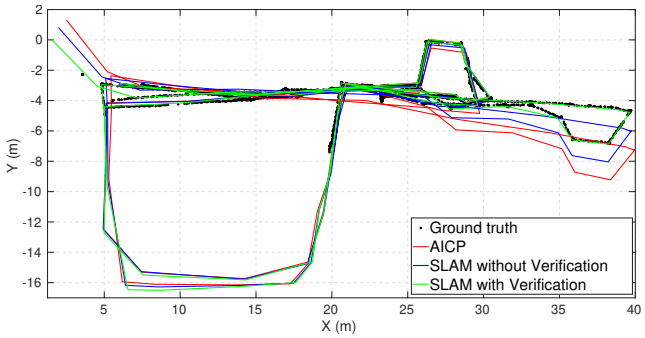


Fig. 7: Result of SLAM system with verification enabled (left). Estimated trajectories of algorithm variants versus ground truth (right).

around the same area. The geometric loop closures iterated over the nearest N clouds, based on a radius; the covariance and distance travelled caused it to slow down. Similarly, the verification method needs to iterate over large proportion of the point clouds in the same area, affecting the real time operation.

Instead, the learning loop closure proposal scales better with map size. It compares low dimensional feature descriptor vectors, which is much faster than the thousands of data points in Euclidean space.

C. Indoor and Outdoor Experiments

To evaluate the complete SLAM system, the robot walked indoor and outdoor along trajectories with the length of about 100 m and 250 m, respectively. Each experiment lasted about 45 minutes. Fig. 1 (Bottom) and Fig. 4 illustrate the test locations: industrial buildings. To evaluate mapping accuracy, we compared our SLAM system, AICP, and the baseline legged odometry (TSIF) using ground truth. As shown in Fig. 4, we used a Leica TS16 to automatically track a 360° prism rigidly mounted on top of the robot. This way, we could record the robot's position with millimeter accuracy at about 7 Hz (when in line of sight).

For evaluation metrics, we use Relative Pose Error (RPE) and Absolute Trajectory Error (ATE) [32]. RPE determines the local accuracy of the trajectory over time. ATE is the RMSE of the Euclidean distance between the estimated trajectory and the ground truth.

As seen in Fig. 7 (right), our SLAM system with verification is almost completely consistent with the ground truth. The verification algorithm approved 27 true-positive loop-closure factors (indicated in red in Fig. 7 (Left)) which were added to the factor graph. Without this verification 38 loop-closures were created, some in error, resulting in an inferior map. Fig. 1 (Top) also evidences the global consistency of the map. However, there is no ground truth available for our indoor experiment for evaluation of the trajectory.

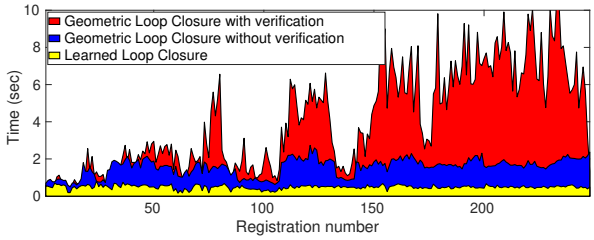


Fig. 8: Computation times of the considered loop closure methods.

Method	Translational Error (RMSE) (m)	Heading Error (RMSE) (deg)	Relative Pose Error (m)
SLAM with Verification	**0.06**	N/A	**0.090**
SLAM without Verification	0.23	1.6840	0.640
AICP (LiDAR odometry)	0.62	3.1950	1.310
TSIF (legged odometry)	5.40	36.799	13.64

TABLE II: Comparison of the localization accuracy for the different approaches.

Tab. II reports SLAM results with and without verification compared to the AICP LiDAR odometry and the TSIF legged state estimator. As the Leica TS16 does not provide rotational estimates, we took the best performing method - SLAM with verification - and compared the rest of the trajectories to it with the ATE metric. Based on this experiment, the drift of SLAM with verification is less than 0.07%, satisfying many location-based tasks of the robot.

D. Experiments on the ANYmal

In a final experiment, we tested the SLAM system online on the ANYmal. After building a map with several loops (while teleoperated), we queried a path back to the operator station. Using the Dijkstra's algorithm [33], the shortest path was created using the factor graph. As each edge has previously been traverse, following the return trajectory returned the robot to the starting location. The supplementary video demonstrates the experiment.

V. Conclusion and Future Work

This paper presented an accurate and robust LiDAR-SLAM system on a resource constrained legged robot using a factor graph-based optimization. We introduced an improved registration verification algorithm capable of running in real time. In addition, we leveraged a state-of-the-art learned loop closure detector which is sufficiently efficient to run online and had significant viewpoint robustness. We examined our system in indoor and outdoor industrial environments with a final demonstration showing online operation of the system on our robot. In future work, we will speed up our ICP registration to increase the update frequency of our SLAM system and examine our system in more varied scenarios.

VI. Acknowledgement

We would like to thank our colleagues in ORI, in particular Simona Nobili, for their help in this work. This research was supported by the Innovate UK-funded ORCA Robotics Hub (EP/R026173/1) and the EU H2020 Project MEMMO. M. Fallon is supported by a Royal Society University Research Fellowship.

REFERENCES

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *TRO*, 2015.

[2] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *ECCV*, 2014.

[3] H. Porav, W. Maddern, and P. Newman, "Adversarial training for adverse conditions: Robust metric localisation using appearance transfer," in *ICRA*, 2018.

[4] S. Nobili, R. Scona, M. Caravagna, and M. Fallon, "Overlap-based ICP tuning for robust localization of a humanoid robot," in *ICRA*, 2017.

[5] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State estimation for legged robots-consistent fusion of leg kinematics and IMU," in *RSS*, 2013.

[6] O. Stasse, A. Davison, R. Sellaouti, and K. Yokoi, "Real-time 3D slam for humanoid robot considering pattern generator information," in *IROS*, 2006.

[7] N. Kwak, O. Stasse, T. Foissotte, and K. Yokoi, "3D grid and particle based slam for a humanoid robot," in *Humanoids*, 2009.

[8] G. Oriolo, A. Paolillo, L. Rosa, and M. Vendittelli, "Vision-based odometric localization for humanoids using a kinematic EKF," in *Humanoids*, 2012.

[9] R. Wagner, U. Frese, and B. Bäuml, "Graph SLAM with signed distance function maps on a humanoid robot," in *IROS*, 2014.

[10] R. Scona, S. Nobili, Y. R. Petillot, and M. Fallon, "Direct visual SLAM fusing proprioception for a humanoid robot," in *IROS*, 2017.

[11] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking." in *ISMAR*, 2011.

[12] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM without a pose graph." RSS, 2015.

[13] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–606.

[14] M. Milford and G. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," in *ICRA*, 2012.

[15] H.-W. Park, P. Wensing, and S. Kim, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," in *RSS*, 2015.

[16] S. Nobili, M. Camurri, V. Barasuol, M. Focchi, D. Caldwell, C. Semini, and M. Fallon, "Heterogeneous sensor fusion for accurate state estimation of dynamic legged robots," in *RSS*, 2017.

[17] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in real-time." in *RSS*, 2014.

[18] J.-E. Deschaud, "Imls-slam: scan-to-model matching based on 3d data," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2480–2485.

[19] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain," in *IROS*, 2018.

[20] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the bayes tree," *IJRR*, 2012.

[21] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments." in *Robotics: Science and Systems*, 2018.

[22] R. Dubé, A. Gawel, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "An online multi-robot SLAM system for 3D LiDARS," in *IROS*, 2017.

[23] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "Segmatch: Segment based place recognition in 3D point clouds," in *ICRA*, 2017.

[24] M. Bloesch, M. Burri, H. Sommer, R. Siegwart, and M. Hutter, "The two-state implicit filter recursive estimation for mobile robots," *RAL*, 2017.

[25] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing icp variants on real-world data sets," *Autonomous Robots*, 2013.

[26] G. Tinchev, A. Penate-Sanchez, and M. Fallon, "Learning to see the wood for the trees: Deep laser localization in urban and natural environments on a CPU," *RAL*, 2019.

[27] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *NIPS*, 2018.

[28] S. Nobili, G. Tinchev, and M. Fallon, "Predicting alignment risk to prevent localization failure," in *ICRA*, 2018.

[29] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, and M. Bloesch, "Anymal - a highly mobile and dynamic quadrupedal robot," in *IROS*, 2016.

[30] E. Cheney and D. Kincaid, *Numerical mathematics and computing*. Cengage Learning, 2012.

[31] J. Zhang, M. Kaess, and S. Singh, "On degeneracy of optimization-based state estimation problems," in *ICRA*, 2016.

[32] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IROS*, 2012.

[33] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, 1959.