Automating Node Pruning for LiDAR-based Topometric Maps in the context of Teach-and-Repeat

David Landry, Philippe Giguère
Department of computer sciences and software engineering
Université Laval
Québec, Canada
Email: davidlandry93@gmail.com, philippe.giguere@ift.ulaval.ca

***Abstract*—Maps are a vital component in autonomous mobile systems. However, they can be computationally expensive to build, particularly in the case of globally-consistent ones. To circumvent this issue, topometric approaches have been proposed, especially in the context of the *Teach-and-Repeat* paradigm. These topometric maps are graphs, where nodes are local maps and edges are the known transform between these nodes. When building these topometric maps, one important question is how far apart should these local maps be collected in the environment. On one hand, a densely sampled topometric map will be more robust, but at the cost of an increased database size. On the other hand, fewer local maps mean a more efficient topometric map. Traditional approaches record these local maps at a fixed, regular intervals. In this paper, we propose an offline algorithm that automatically prunes nodes in a way that offers some empirical guarantees on localization errors. In particular, our approach is based on first collecting a densely sampled topometric map, then finding a minimal subset of nodes using a cost-based approach via Dijkstra's Algorithm. Offline experiments on three datasets confirmed the ability of our approach to minimize the size of topometric maps, and showed that the size of the map will vary given a certain demanded localization error tolerance and environment complexity.**

I. INTRODUCTION AND PREVIOUS WORK

Teach-and-Repeat is a popular and efficient paradigm to execute displacement tasks where it is possible to have an operator show to a robot how to accomplish it beforehand. This way, the perceptual capabilities of the mobile system can be greatly lowered. The *Teach-and-Repeat* paradigm does not rely on a particular sensing modality; indeed, it has been implemented using stereo cameras [1] and LiDAR sensors [2], and used in challenging situations such as underground mines [3] or with a flying quadrotor [4].

At the heart of all *Teach-and-Repeat* approaches is the concept of topometric maps. They combine locally metric maps that are perceptually linked to the localisation sensing capabilities (range and modality) within a more abstract global, topological map (Kuipers *et al.* [5]). The Atlas framework [6] was one of the first to demonstrate that such an approach could answer the problem of large scale mapping. They achieved this by combining smaller, locally metric maps into a larger topological map that contained the relative transforms between its local maps. This topometric paradigm has also been employed for the problem of multi-robot mapping and exploration of an unknown environment, to alleviate the problem of place recognition and loop closure by using the robot themselves as robust landmarks [7]. Loosely related to this concept of topometric map for 3D sensors, Steder *et al.* [8] have proposed place recognition from range images, a form of 2½D local mapping, where scans were taken at fixed intervals, as in most *Teach-and-Repeat* implementations.

The idea of performing some kind of optimization within the paradigm of *Teach-and-Repeat* is not new. For example, [9] presents an approach (*teach-optimize-repeat*) that tries to optimize the smoothness of the trajectory while simultaneously minimizing its duration. Optimisation of the speed schedule [10] or of the linear controller performing the *repeat* phase [11] have also been explored. Contrary to our approach, none of them try to minimize the number of nodes within the topometric map.

The concept of selecting nodes in topometric maps is also found in keyframe-based visual mapping techniques. As they try to operate in real-time, the triggering of node creation usually relies on simple heuristics. This kind of approach is found for example in the well-know PTAM algorithm [12]. Another keyframe-based approach, RTAB-MAP [13], uses an heuristic based on the difference in appearances: a new node is added when the ratio of visual words detected from the previous node falls below a threshold. No map curating is performed afterwards. Node creation decisions can also be found in some work related to place recognition. In [14] for example, they perform topological mapping using vision, with an Incremental Spectral Clustering. Their goal is to find a cluster in visual appearance space that would be representative of a region, with such a cluster becoming a node in the map. They noted, unsurprisingly, that their algorithm generated more nodes in narrow indoor areas than in larger, open outdoor ones. In some sense, an autonomous topometric mapping algorithm should behave similarly, as localisation capability is somewhat related to place recognition capabilities.

One of the questions that arise when implementing *Teach-*

and-Repeat algorithms is how sparsely should the nodes in the topometric map be placed. A very frequent sampling, with a large number of local maps to localize against during the *repeat* run, will provide more robustness against localisation failures. However this robustness comes at the cost of a very large database (map) size, making it a storage burden. Moreover, if place recognition is added later on, it will make detecting loop closures more expensive by increasing the search space, although this can be mitigated by feature-words and inverted index approaches such as FAB-MAP [15]. Anecdotal evidence shows that the density of sampling of local maps should be intimately related to the environment itself. For example, in [8], place recognition recall rates are different for the two datasets they used (Hanover vs. Freiburg.) For Freiburg dataset (a more open-space), the distance corresponding to a recall of 50% is around 25 m, while for the Hanover dataset it is at around 12 m.

Current *Teach-and-Repeat* approaches leave it to the operator to decide the distance between the nodes in the topometric map. For example, in [1], [2], the robot records a new local map at regular intervals during the *teach* phase, based on the robots odometry. This approach does not take into account the possibly varying complexity of an environment. Consequently, the operator has to be aware of the underpinning localization technique in order to make an educated guess for the sampling distance parameter, or resort to trial-and-error.

Our proposed approach borrows some conceptual elements from Churchill *et al.* [16]. In their work, they augment the traditional topometric maps used within the *Teach-and-Repeat* paradigm with a spatial model of the localizer performance. They aim at estimating how far away can the robot deviate from the taught trajectory before the localizer fails, thus finding what they call a *localization envelop*. In our approach, we will employ the ability for a localizer to perform within an error bound to determine which subset of local maps is sufficient to perform localization on the taught trajectory. This way, we can prune redundant nodes without sacrificing too much of the localization robustness during the *repeat* phase. More importantly, the operator now simply has to provide an error tolerance on localization, instead of choosing at what distance should the local maps be selected in a particular environment. We believe it is a step forward in having a more intuitive and natural way of parameterizing the topometric mapping process.

II. PROBLEM DEFINITION

We have a topometric map that has been recorded during a *teach* phase at fixed, small intervals, containing a set $N = \{n_1, n_2, ..., n_m\}$ of m interconnected nodes. Each of these nodes n_i is a local map in the form of a 3D point cloud z_i that has been acquired at the origin of the local map by a LiDAR. The geometric transformation (rotation, translation) between the frame of reference of consecutive nodes n_i and n_{i+1} is ${}^{\ i}_{i+1}\mathbf{T}$, and is stored in the map as well, as the edges between consecutive nodes.

Our goal is to find the smallest subset of nodes of N such that we can still perform the *repeat* phase with some high degree of confidence for a region Q. Q itself contains the taught trajectory and a user-defined deviation around it. This region is assumed to be traversable by the robot, and is used to take into account the unavoidable motion errors. In order to be able to perform the *repeat*, the robot must be able to localize at any point of this region Q. In some sense, Q is loosely related to the *localization envelop* of Churchill *et al.* [16]. Our problem can be described more formally as finding a subset $S \subseteq N$ of nodes s_i such that $\exists s_i \in S$ such that we have $\texttt{LOCALISES}(s_i, q)$, $\forall q \in Q$. We can think of the $\texttt{LOCALISES}$ function as an indication that it is possible for a robot receiving the sensor reading z_q during a repeat phase to localize using a local map stored in S.

Several problems arise from this purely theoretical formulation. First, the $\texttt{LOCALISES}$ function is more or less impossible to compute reliably in our case, we will approximate it by looking at the final error of a localization algorithm, such as Iterative Closest Point (ICP.) Second, testing this function for all possible q's is not feasible. Indeed, [16] performs a limited number of samples during *repeat* phases, and uses these samples to update a Gaussian Process to estimate the *localization envelop*. In our case, we will further simplify the procedure by re-using scans taken during the teaching phase as samples of q. This increases the risk that Q is not properly sampled (as we do not go technically off the taught path). However, it will allow our approach to be able to perform pruning offline, without the need for an online optimization phase. Finally, we will assume that we always use the local map in the current node for localization, and that node are selected in increasing order (as is typically done in a *Teach-and-Repeat* scenario).

This pruning is a trade-off between robustness of localization and storage requirements for the map. As we depart further from a node n_i, the likelihood for a localizer to fail in its local map generally increases with the distance from the origin of that map (where the pointcloud z_i was acquired.) By reducing the number of nodes, we therefore increase the average distance between a sensor reading z_t and the origin of the currently active local map. On the other hand, the pruning decreases the storage requirements for the topometric map. As each node can contain a full 3D scan, the memory savings can be substantial for LiDAR-based methods. For example, we employed a Velodyne HDL-32E LiDAR in our experiments, and each local map needed approximately 1 MB of storage. Reducing the number of nodes thus reduces the storing requirements of the map, as well as speeding up the loading of these maps from disk.

III. OUR APPROACH

This section presents in greater details our approach. In particular, we will first describe how we approximate `LOCALISES` with ICP, and how we approximate the sampling of the region Q. We then explain how the actual pruning can be reframed as a shortest-path problem in a graph, and solved efficiently with Dijkstra's algorithm.

A. Simulating the `LOCALISES` function

In our implementation of *Teach-and-Repeat*, we use the Iterative Closest Point (ICP) algorithm as our localizer. Therefore, it is only natural to approximate the $\texttt{LOCALISES}(n_k, n_l)$ function by the use of ICP itself. We will declare that if ICP does not localize the node n_l, using n_k as a local map within a certain positional error ϵ_{pos} or angular error $\epsilon_{angular}$, then `LOCALISES` fails. These ϵ parameters are provided by the user. We argue that since they are physical quantities (distances or angle), they provide for an intuitive parameter to set. This definition of the `LOCALISES` function will cause our algorithm to adapt itself to the environment of the map. As ICP tends to fail in complex and narrow areas or when the matched clouds are rotated with respect to one another, this approach will tend to detect such situations and keep the related nodes, thus preserving the general robustness of the map.

In an ideal situation, one could perform this pruning online and interactively, in a *Teach-optimize-repeat* manner. For example, in [16] the *localisation envelop* is identified through an active-sampling approach: the vehicle is progressively driven off the taught trajectory during repeat. However, this kind of approach can be time consuming. Therefore, it is highly desirable to be able to perform node pruning offline. To achieve this, we will evaluate the quality of a node n_k in terms of ability to localize by using the LiDAR scan z_l stored as local maps subsequent nodes n_l $(l > k)$. Note that all these nodes were acquired during the *teach* phase, and are therefore part of the initial map N. We evaluate this localisation ability by testing $\texttt{LOCALISES}(n_k, n_l)$, which tell us if node n_k (with local map z_k) can be used as a reference point when we receive a new LiDAR scan (simulated by z_l) during the *repeat* phase. In our context, it boils down to knowing if the ICP algorithm is robust, given these inputs. Our core idea here is to affirm that we can be confident in localizing if ICP can converge reliably, despite errors in initial estimation of the transform between the two input scans. We will thus approximate `LOCALISES` by running the ICP algorithm on the given point cloud pair (z_k, z_l), with a number of different initial transformation disturbances E. If the ICP succeeds every time (*i.e.* is within ϵ_{pos} and $\epsilon_{angular}$), then we declare that we are confident that the point cloud z_k pair is good for localizing near z_l.

More formally, we derive $\texttt{LOCALISES}(n_k, n_l)$ by executing the following steps. First we will sample the space Q by using P disturbances transformation matrix $\mathbf{E}_p$ that induces small spatial (Δx, Δy) and angular ($\Delta\theta$) displacements near a fixed location. These disturbances are chosen so as to be regularly-spaced on the surface of a 3D ellipsoid. A 2D simplification of this is illustrated in Fig. 1. For performances reasons, we did not sample any point inside the ellipsoid, as the ICP localization is generally more successful with better initial estimates. The length of the semi-axes of the ellipsoid is determined through three user-provided parameters, x, y and θ, allowing the user to choose what kind of disturbances he wants the map to be tolerant to. The ellipsoid was parameterized as follows in our experiments:

$$\begin{aligned}
\Delta x &= x \cos u \sin v \\
\Delta y &= y \cos u \cos b \\
\Delta\theta &= \theta \cos v
\end{aligned} \tag{1}$$

for $u \in [0, 2\pi), v \in [0, \pi]$. In our implementation, this ellipse was sampled for $P = 25$ different disturbances $\mathbf{E}_p$ per execution of the `LOCALISES` function.

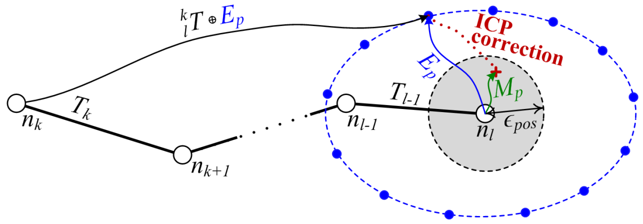

Figure 1: A simplified view of the `LOCALISES` predicate, for the case where the angular disturbance is null. The blue dots represent the P samples taken on the disturbances ellipsoid, which are our sampling of space Q near node n_l. Note that in our approach, this ellipse is 3-dimensional, the 3^{rd} dimension being related to the angular disturbance. Matrix $\mathbf{M}_p$ represents the transformation error after ICP.

Each of these P disturbances $\mathbf{E}_p$ will then be used to create a test for the `LOCALISES` function. As we mentioned earlier, we will use ICP, which we define as the following function:

$$\texttt{ICP}(z_x, z_y, \mathbf{T}_{init}) \tag{2}$$

where z_x and z_y are the input point clouds and $\mathbf{T}_{init}$ is the initial transformation guess between these two point clouds. This function returns a transformation matrix that (if the ICP succeeds) moves the frame of z_x (sufficiently close) to the frame of z_y.

Now that we picked a sample in Q, we need a reference to which we can compare the result of the disturbed ICP. When testing for a pair of nodes n_k and n_l, we compute an undisturbed location estimate, which is simply the relative pose of node n_l in the frame of reference of node n_k:

$${}^k_l\mathbf{T} = {}^{k}_{k+1}\mathbf{T} \oplus {}^{k+1}_{k+2}\mathbf{T} \oplus \cdots \oplus {}^{l-1}_{l}\mathbf{T} \tag{3}$$

where $\oplus$ is the usual compounding operator [17].

We will collect the difference between the undisturbed estimate ${}^k_l\mathbf{T}$ and the result of the ICP with a disturbed initial estimate $\mathbf{E}_p \oplus {}^k_l\mathbf{T}$ for each $\mathbf{E}_p$. The result is stored in the form of a transformation matrix $\mathbf{M}_p$:

$$\mathbf{M}_p = \texttt{ICP}(z_k, z_l, \mathbf{E}_p \oplus {}^k_l\mathbf{T}) \ominus {}^k_l\mathbf{T} \tag{4}$$

where $\ominus$ is the difference operation for transformations (equivalent to composing the first transformation with the inverse of the second), and z_i is the local scan associated with node n_i. We extract the translation vector and rotation matrix from every error matrix $\mathbf{M}_p$.

$$\mathbf{M}_p = \begin{bmatrix} \mathbf{R}_p & \mathbf{t}_p \\ \mathbf{0}^t & 1 \end{bmatrix}. \tag{5}$$

Then we compute the ICP errors for the P disturbed samples in terms of translation (e_t) and rotation (e_r) errors:

$$e_{t,p} = \|\mathbf{t}_p\| \tag{6}$$

$$e_{r,p} = \arccos\left(\frac{\text{trace}(\mathbf{R}_p) - 1}{2}\right). \tag{7}$$

For `LOCALISES` to be considered successful, these errors for all the P disturbances must be smaller than the user-provided thresholds ϵ_{pos} and ϵ_{angle}:

$$\begin{aligned} &\texttt{LOCALISES}(n_k, n_l) \Leftrightarrow \\ &(e_{t,p} < \epsilon_{pos} \wedge e_{r,p} < \epsilon_{angle} \quad \forall p \in \{1, 2, \ldots, P\}). \end{aligned} \tag{8}$$

For instance in Fig. 1 we consider that ICP (showed as the red dotted line) is successful for sample p if ICP is able to compensate a disturbed initial estimate (${}^k_l\mathbf{T} \oplus \mathbf{E}_p$) so that its final transformation is located in the greyed region within ϵ_{pos} close from the undisturbed position ${}^k_l\mathbf{T}$. `LOCALISES` is considered successful if all disturbances $\mathbf{E}_p$ are successfully compensated by ICP.

B. *Building and optimizing the localisability graph*

The localisability graph is a graph where the nodes are the same as our initial map N. The edges now represent the result of the `LOCALISES` function between two local maps: if `LOCALISES`(n_i, n_j) is successful, then an edge between nodes n_i and n_j is added in the graph, with a fixed cost of 1. We assume that there is an edge between every pair of consecutive nodes, which translates to assuming that the robot is able to localize in Q using the original, unoptimized map. For performance reasons, we do not run the `LOCALISES` function on a pair of nodes (n_i, n_k) if `LOCALISES` failed for a pair *i.e* (n_i, n_j) such that $j < k$. Here we simply assume that all nodes beyond n_j will fail for n_i, as ICP tend to perform less and less well as distance increases. Moreover, this will make our approach conservative in terms of pruning, thus it is not detrimental from a robustness point of view.

Once we have the localisability graph in hand, we need to find the best subset S of nodes such that we can reliably localise. To this effect, we use Dijkstra's algorithm for finding the shortest path in a graph, with the first node as a starting point. By finding the path that has the fewest edges (and consequently the fewest nodes) from the beginning of the map to the end, we therefore find a subset of minimum size such that we can still localize within this map, according to our approximation of `LOCALISES` and our sampling strategy. The nodes that are traversed by this path are the nodes in our optimized map. The complexity of Dijkstra is $O(l \log m)$ (where l is the number of edges in the localisability graph) in this case. This path is orders of magnitude easier to compute that the `LOCALISES` function on the successive pairs of nodes, and thus is not a burden on a computing time level. It also keeps us from having to define a heuristic for the search.

Throughout this paper we assume that the robot will always use a local map *behind* him to localize the current reading during a *repeat* run. Consequently we only have to compute `LOCALISES`(n_i, n_j) for $j > i$. Making supplementary assumptions about the capabilities of the *repeat* software could lead to a better optimization.

The source code of the optimization framework is available at http://bit.ly/1oKsSxd.

IV. EXPERIMENTS

A. *Datasets collection*

We collected three datasets using a Clearpath Husky A200 equipped with a Velodyne HDL-32e LiDAR sensor. The LiDAR scans are used both as local maps in nodes, and in computing the relative transforms ${}^i_j\mathbf{T}$ between each node via ICP during the *teach* passes. This approach is more precise than relying on the odometry of the wheels in places with uneven ground with poor traction, but might cause some discontinuities in the robot trajectory. The first dataset, `terasse`, is a short run on paving stones on Laval University's campus in a relatively open area. As it was a busy summer day, passer-by that were curious about the robot introduced dynamic outliers in the point clouds. The `forest` dataset is a short run in some forested area on campus, making the environment highly unstructured and complex, as well as somewhat narrow due to the presence of foliage along the walking path. Finally, the third dataset `hallway` mapped an indoor hallway as the robot went in a straight line. We recorded a higher density of nodes for this dataset, to specifically verify that pruning of nodes will be more aggressive with our algorithm than with the two other testsets. The *teach* passes were executed using a home-grown *Teach-and-Repeat* software.

We used `libpointmatcher` [18] as our ICP engine throughout this work. The parameters given to the ICP engine were the same as the ones we used in our *Teach-and-Repeat* implementation, and contained (among other

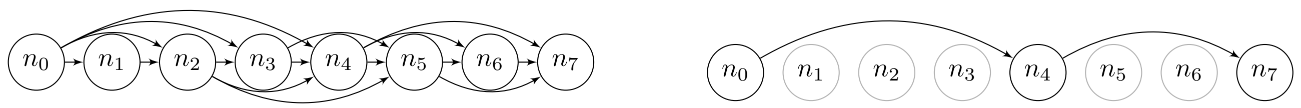

Figure 2: The localisability graph, before and after running Dijkstra's algorithm. Here, every node is from the unoptimized *Teach-and-Repeat* map. An edge between two nodes indicates that the `LOCALISES` function succeeded from the first node to the second. In this example, `LOCALISES` succeeded for the pair (n_2, n_5) but failed for (n_1, n_3). By running Dijkstra's algorithm on this graph (with every edge having a cost of 1), we can find the shortest path from the first node to the last. Our optimized map will be comprised of the nodes used by this path, as shown on the right.

configurations) random subsamplings such that 15 % of the points of the new scan and 25 % of the points of the local map would be kept during ICP operations.

B. *Results of the optimization*

Table I presents the results of applying our algorithm to the three previously described datasets. The parameters of the optimization framework were $x = 1.5\,\text{m}$, $y = 0.75\,\text{m}$ and $\theta = 0.18\,\text{rad}$. The convergence threshold were $\epsilon_{pos} = 0.2\,\text{m}$ and $\epsilon_{angle} = 0.09\,\text{rad}$. The length of the trajectories were approximated with the sum of the distances between successive nodes in the map.

The results are highly dependent on the geometry of the environment; sharp turns in the robot trajectory are difficult to optimize, as are complex and narrow environments. Moreover, we found that the parameters do not need to change a lot from one environment to another, despite wild variations in environment complexity and type (indoor vs. outdoor.) This seems to indicate that the parameters are somewhat independent of the environment, as we would hope. Figs. 3 and 4 show which nodes were kept during the automatic pruning for the `hallway` and `terrasse`, respectively. Fig. 5 shows the evolution of the average distance between successive nodes, for each dataset.

C. *Behaviour of convergence of ICP*

Our algorithm relies heavily on the fact that we expect ICP to fail when nodes are too far apart for localisation purposes. To validate this concept empirically, we explored the behaviour of the translation and rotation errors e_t and e_r, after ICP adjustment, as nodes are further apart. To do so, we used the local map of a fixed node n_i, and performed ICP on subsequent nodes, *i.e.* on pairs (n_i, n_{i+1}), (n_i, n_{i+2}), …, (n_i, n_{i+N}) where N is the total number of nodes in the topometric map. To estimate the distribution of these errors, these tests were repeated 50 times, as the ICP engine performs random subsampling. The error was computed the same way it was in Eq. (6) and Eq. (7), except that the perturbation $\mathbf{E}$ was set to the identity matrix (no perturbation). This validation was done on the `terasse` dataset, with the results shown in Fig. 6. What can be interpreted from these graphics is that ICP tended to have a smooth and gentle degradation over shorter distances, but

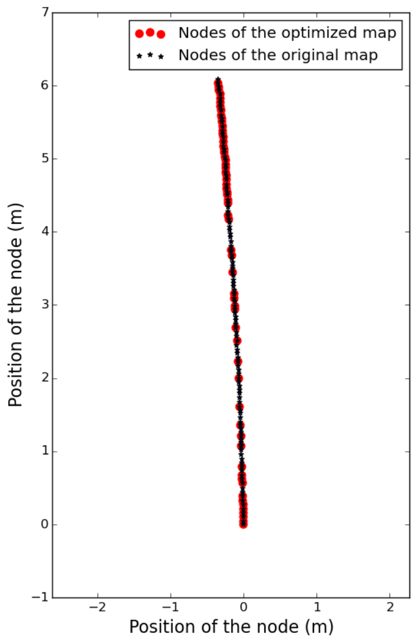

Figure 3: The result of the optimization algorithm on the `hallway` dataset. We notice that the optimization is highly affected by the geometry of the environment. In this case, it is possible to optimize away a large number of nodes towards the middle of the map. The upper extremity requires more nodes, due to the occlusions around open doors located in that region.

displayed a phase transition by abruptly failing beyond a certain distance. Having such a drastic increase in error (and variance on this error) is highly desirable for our algorithm, as it makes the selection of error threshold ϵ_{pos} and ϵ_{angle} simpler and more robust.

D. *Length of the semi-axes of the disturbance ellipse*

Parameters x, y (translations) and θ (rotation) are the lengths of the semi-axes of the disturbance ellipse and thus determine how robust we want the map to be to those errors. Indeed, the larger this ellipse, the more robust to

Dataset	Trajectory Length	N. of nodes	N. of nodes (optimized)	fraction kept
Units	m	-	-	(%)
`hallway`	6	117	53	45.3
`terasse`	64	304	211	69.4
`forest`	17	62	49	60.9

Table I: Result of the optimization on different datasets.

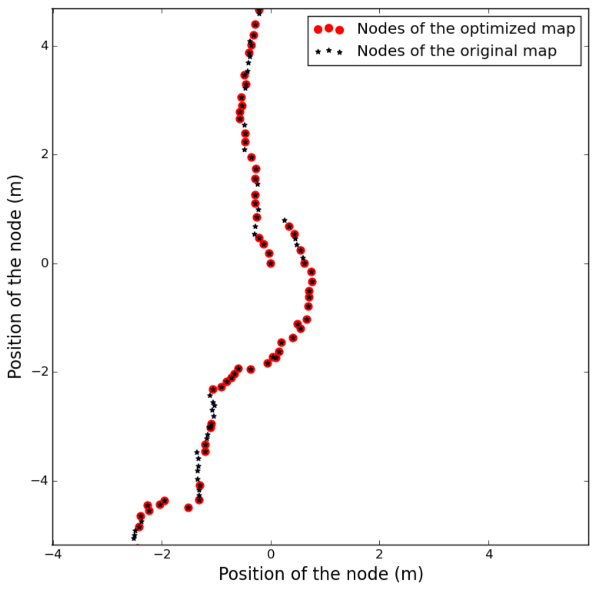

Figure 4: The result of the optimization algorithm on the `terasse` data set. The jumps in the position of the robot are due to the ICP odometry failing to find proper transformations between successive local maps.

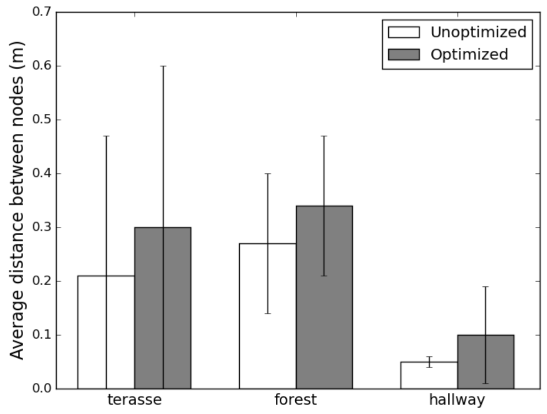

Figure 5: Average distance between successive nodes in each dataset. In every case, this distance between successive nodes increased after optimisation, as expected.

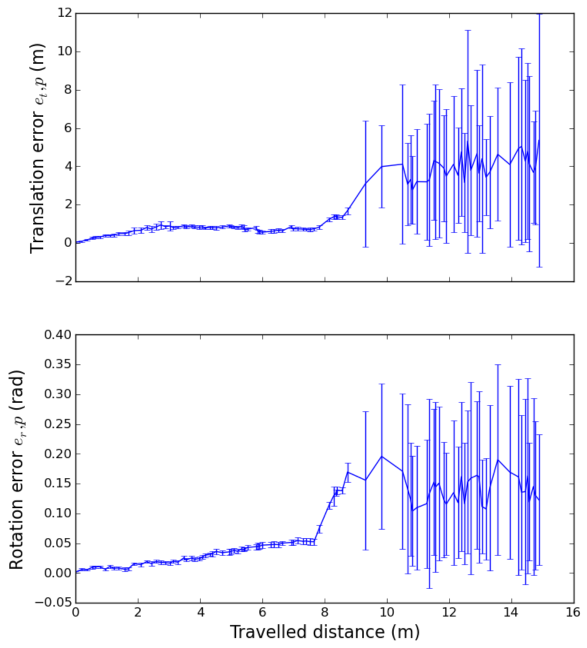

Figure 6: Mean translation and rotation errors according to the travelled distance. This data is from the `terrasse` dataset. The slow growth in error we notice at first can be attributed to the accumulated errors of the odometry when recording the point clouds. The very large growth in variance and error that happens afterwards can be attributed to the ICP runs converging in local minima that are both numerous and far from the true position.

disturbance the final map should be. This makes it less probable to remove nodes, but yields map in which we can still be confident that localisation can be achieved reliably. We tested our algorithm for different values of these parameters to estimate the sensitivity of our algorithm to their values. As can be seen from Fig. 7, the algorithm is more sensitive to angular tolerance that spatial ones. Second, we can see that the spatial parameters x and y do not have a significant impact on the result of the optimization, which could indicate that the ease of localization is more determined by the properties of the environment itself, as we are hoping.

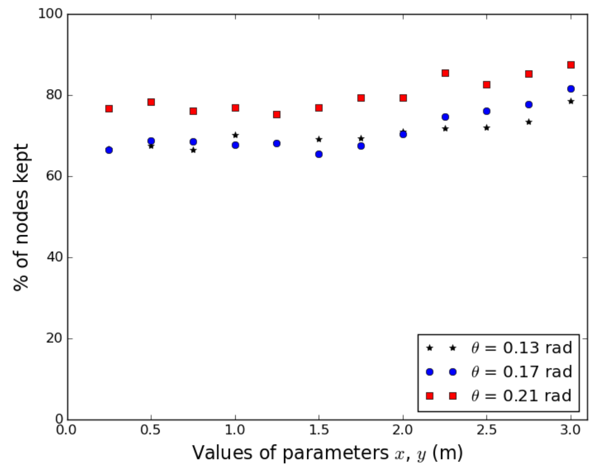

Figure 7: Percentage of nodes kept as a function of the size of the disturbance ellipse used to sample $\mathbf{E}_p$. Unexpectedly, it becomes harder to prune a map as the demanded tolerance to error grows (larger size of the disturbance ellipse). However, the lengths x and y of the semi-axes of the disturbance ellipsoid for physical distance have little to no effect on the end results, while the pruning is more sensitive to angular displacements θ. Each dot represents an optimization run on the `terasse` dataset where $x = y$, with θ specified in the legend.

E. Effect of the ϵ_{pos} and ϵ_{angle} parameters

To better comprehend the effect of the environment on the behaviour of optimization, we ran the optimization framework on our three datasets with increasing values of ϵ_{pos}. The results can be seen on Fig. 8. The optimization runs were done with $x = y = 1\,\text{m}$, $\theta = 0.17\,\text{rad}$ and $\epsilon_{angle} = 0.09\,\text{rad}$.

V. FUTURE WORK AND CONCLUSION

This paper introduced an offline algorithm that can automatically reduce the number of nodes needed in a topometric map for use in *Teach-and-Repeat* implementations. Our map curating algorithm is based on computing the localisation ability of candidate local maps, and using subsequent nodes as test scans. The results of these computations are used to prune nodes in the topometric map until it is as small as possible so that we can execute a *repeat* run with a demanded tolerance to error provided by the user. Importantly, the algorithm has a linear complexity with the size of the map for reasonable environments, as it is based on Dijkstra's algorithm. Our approach was tested with success on three datasets (two outdoors, one indoor) using 3D point clouds from a LiDAR sensor, with ICP used for localisation. Experiments confirmed that the algorithm is indeed able to optimize topometric maps, and that the density of nodes depends on the type of environment and less on the spatial tolerance parameters.

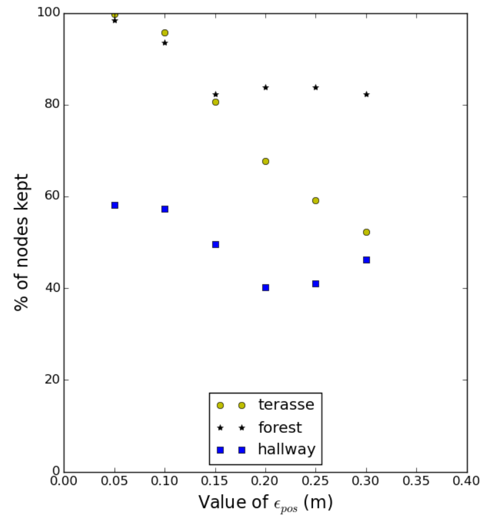

Figure 8: The effect of the parameter ϵ_{pos} in different dataset. The results vary wildly from one environment to the other, indicating once again that the aspect of the environment is a very important factor when deciding how frequently it should be sampled in a *Teach-and-Repeat* map.

We believe that this work can be extended in different ways. First, the current computing bottleneck is the `LOCALISES` predicates, which requires the computation of P ICP's along the disturbance ellipse. However, as as each of these P ICP's can be run independently, we could thus parallelise `LOCALISES` over many CPU cores or rely on a ICP library leveraging GPUs. Another possibility would be to reduce the number of calls to ICP, by using an active learning approach based on Gaussian Processes [16], instead of a regular sampling.

Finally, this work could also be improved by making supplementary assumptions about the *repeat* implementation. Throughout this paper, we assumed that the robot can only use reference points that are *behind* itself during the *repeat* run. By supposing that the robot can use local maps in both directions, the maps could be optimized even further.

REFERENCES

[1] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, vol. 27, no. 5, pp. 534–560, 2010.

[2] C. Sprunk, G. D. Tipaldi, A. Cherubini, and W. Burgard, "Lidar-based teach-and-repeat of mobile robot trajectories," in *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, 2013, pp. 3144–3149.

[3] J. Marshall, T. Barfoot, and J. Larsson, "Autonomous underground tramming for center-articulated vehicles," *Journal of Field Robotics*, vol. 25, no. 6-7, pp. 400–421, 2008.

[4] A. Pfrunder, A. P. Schoellig, and T. D. Barfoot, "A proof-of-concept demonstration of visual teach and repeat on a quadrocopter using an altitude sensor and a monocular camera," in *Proc. of the Conference on Computer and Robot Vision (CRV)*, 2014, pp. 238–245.

[5] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli, "Local metrical and global topological maps in the hybrid spatial semantic hierarchy," in *Robotics and Automation, IEEE International Conference on*, vol. 5, 2004, pp. 4845–4851.

[6] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller, "An atlas framework for scalable mapping," in *Robotics and Automation, IEEE International Conference on*, 2003, pp. 1899–1906.

[7] A. Howard, G. Sukhatme, and M. Mataric, "Multi-robot simultaneous localization and mapping using manifold representations," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1360–1369, Jul. 2006.

[8] B. Steder, G. Grisetti, and W. Burgard, "Robust place recognition for 3d range data based on point features," in *Robotics and Automation, IEEE International Conference on*, 2010, pp. 1400–1405.

[9] M. Mazuran, C. Sprunk, W. Burgard, and G. Tipaldi, "Lextor: Lexicographic teach optimize and repeat based on user preferences," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2015, pp. 2780–2786.

[10] C. J. Ostafew, A. P. Schoellig, T. D. Barfoot, and J. Collier, "Speed daemon: Experience-based mobile robot speed scheduling," in *Proc. of the Conference on Computer and Robot Vision (CRV)*, 2014, pp. 56–62.

[11] C. J. Ostafew, J. Collier, A. P. Schoellig, and T. D. Barfoot, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking," *Journal of Field Robotics*, 2015, To appear.

[12] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality (ISMAR), 6th IEEE and ACM International Symposium on*, 2007, pp. 225–234.

[13] M. Labbe and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based slam," in *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, 2014, pp. 2661–2666.

[14] C. Valgren and A. Lilienthal, "Incremental spectral clustering and seasons: Appearance-based localization in outdoor environments," in *Robotics and Automation, IEEE International Conference on*, 2008, pp. 1856–1861.

[15] M. Cummins and P. Newman, "Fab-map: probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[16] W. Churchill, C. H. Tong, C. Gurau, I. Posner, and P. Newman, "Know your limits: Embedding localiser performance models in teach and repeat maps," in *Robotics and Automation, IEEE International Conference on*, 2015, pp. 4238–4244.

[17] R. Smith, M. Self, and P. Cheeseman, "Autonomous robot vehicles," in, Springer-Verlag New York, 1990, ch. Estimating Uncertain Spatial Relationships in Robotics, pp. 167–193.

[18] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing icp variants on real-world data sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[19] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[20] S. Simhon and G. Dudek, "A global topological map formed by local metric maps," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, IEEE, vol. 3, 1998, pp. 1708–1714.

[21] T. Daoust, F. Pomerleau, and G. Dudek, "Light at the end of the tunnel: High-speed, lidar-based train localization in challenging underground environments," in *Computer and Robot Vision (CRV), Canadian Conference on*, IEEE, 2016, forthcoming.