EMap: Real-time Terrain Estimation

Jacobus C. Lock, Fanta Camara, and Charles Fox

School of Computer Science, University of Lincoln, UK

Abstract. Terrain mapping has a many use cases in both land surveyance and autonomous vehicles. Popular methods generate occupancy maps over 3D space, which are sub-optimal in outdoor scenarios with large, clear spaces where gaps in LiDAR readings are common. A terrain can instead be modelled as a height map over 2D space which can iteratively be updated with incoming LiDAR data, which simplifies computation and allows missing points to be estimated based on the current terrain estimate. The latter point is of particular interest, since it can reduce the data collection effort required (and its associated costs) and current options are not suitable to real-time operation. In this work, we introduce a new method that is capable of performing such terrain mapping and inferencing tasks in real-time. We evaluate it with a set of mapping scenarios and show it is capable of generating maps with higher accuracy than an OctoMap-based method.

1 Introduction

3D terrain mapping is a core problem in mobile robotics and efficient solutions are crucial to enable robots to explore and perform tasks on various terrains. Terrain data also has offline uses in site surveyance, such as to plan new construction projects, environmental monitoring to improve early interventions for flooding, and determining agricultural subsidy payments dependant on terrain quality. Many terrain mapping solutions currently employed use visual methods that rely on many distinct, clear features and are therefore well-suited for indoor or urban outdoor environments where these features are plentiful. However, such features are much rarer in outdoor off-road environments which are largely flat and bereft of many distinct static features, making the popular vision-based methods less robust in these settings.

A popular approach to the terrain mapping problem uses point clouds captured from a LiDAR sensor and merges them together over time using methods such as OctoMap [1] and UFOMap [2] to form a complete map of the environment. However, these methods become less robust the further away data captured by the LiDAR are from the actual sensor – significant gaps in the terrain estimate are introduced as the gaps between LiDAR hits become larger. This is not a particularly serious issue in populated urban or indoor areas, where large, empty areas at further distances (25m+) are fairly uncommon, but becomes acute in off-road outdoor areas, where large, flat expanses with uneven surfaces are the norm. Currently, this problem is addressed by having the sensor

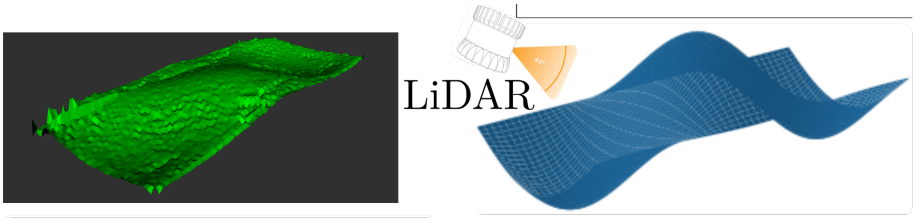

Fig. 1: An example of a surface (left) reconstructed using EMap (right).

pass over the data-sparse areas multiple times to generate additional data with which to fill in the cloud map and increase the terrain estimate's fidelity. Such approaches are appropriate for, and incur little additional cost from, small autonomous robots and vehicles that can make many passes with minimal input from a human operator. However, where large industrial or agricultural robots and vehicles are involved, such a multi-pass approach can become very costly, particularly when human labour is required to monitor the work.

As an example, suppose a LiDAR sensor can detect points up to R meters away. The number of LiDAR hits decays as a function of the distance to the sensor (see Fig. 2 for the sampling density over R for the OS1-64 LiDAR used in this work). If one sets a minimum sampling density requirement for a terrain map, there will exist some range r_m from the sensor that demarcates the maximum distance from the sensor where the sampling density becomes too low and which will require additional passes to sufficiently map. A survey site of width $3r_m$, for example, will then require at least 2 back and forth passes to map with a sufficient number of point data. However, if r_m could be extended, the entire site could be mapped with fewer passes. In many applications, the savings could become substantial when considering the operational costs involved in land surveyance. For example, helicopter and drone-based LiDAR surveyance operations are common and the costs include equipment rent, pilot wage, and fuel, so any reductions in total travel time and distance could lead to a real and substantial economic and environmental impact.

A simple way to increase the effective scanning range r_m is to use a higher-cost LiDAR with an increased sampling density. Alternatively, approximating point positions in and around the gaps in the terrain estimate, based on actual LiDAR inputs surrounding the area, will effectively increase the sampling density and extend r_m. The widely used OctoMap [1] package can be used to create an occupancy map and perform interpolation between the occupied nodes to fill in the unknown nodes, thereby accomplishing the aforementioned approximation task. However, in this paper we show that this OctoMap-based interpolation approach is not robust enough to have a meaningful impact on reducing the surface estimation errors and increase a sensor's effective scanning range, r_m. We therefore introduce a new alternative method, EMap, based on energy minimisation techniques to reduce these errors, thereby increasing r_m and reducing

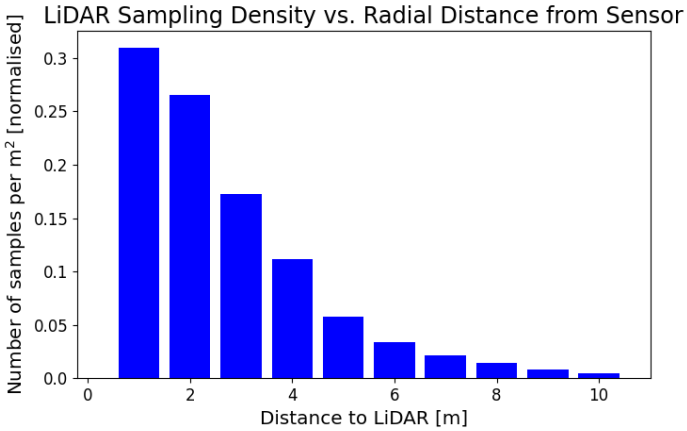

Fig. 2: Number of samples captured by the Ouster OS1 LiDAR used in this work as a function of distance distance to the sensor.

the number of passes needed to generate an accurate terrain map. Fig. 1 shows a sample surface reconstructed using the EMap method.

Many studies regard only online autonomous navigation as the use-case for real-time mapping. However, the parameters and requirements of land surveyance tasks differ significantly and are often not considered. In contrast, we devise a set of utility functions that represent both types of tasks and evaluate our method according to its performance in these simulated tasks. The contributions of this work are therefore as follows: First, a novel approach to terrain mapping, specialised for off-road settings that extends the effective LiDAR mapping range beyond that of OctoMap without significant additional computational cost. Second, a set of utility functions that resemble the priorities and constraints of terrain mapping scenarios.

2 Related work

Computer vision techniques have successfully been used for surface reconstruction for many years and are quite mature at this point. Indeed, stereo vision systems are extensively used in the Mars rovers [3]. However, in the last decade researchers have become increasingly interested in using LiDARs to capture 3D environmental data [4–6]. These sensors offer several advantages over camera-based systems, such as being more robust to different lighting conditions and offering a wider field of view without distortion. Many modern Geographic Information Systems (GIS), which provide a map of a geographic area [7], rely on data captured by some form of LiDAR. Such maps are very useful for environmental and urban planners to predict floods, observe floral patterns over time, or to create relief maps, for example, and form so-called Digital Elevation

Models (DEM). DEMs are often created by merging multiple point cloud observations over a large area together using supplemental localisation data from a GPS, for example. However, these models often break down when different terrain features are captured and are not properly classified [8, 9]. For example, floral canopies may distort the terrain map's height estimate if these are not properly accounted for in the DEM. A smaller scale DEM, restricted to a single terrain type, would not be affected by these issues.

OctoMap [1] and a recent variant, UFOMap [2], offer such a LiDAR-based mapping approach suitable for use by mobile robots and their immediate surroundings. As LiDAR data are received, these systems organise them into an octree [10] structure, which is a highly scalable and optimised data structure for 3D geometric modelling and is very well-suited for high resolution mapping. Octree nodes and leaves are marked as free or occupied according to whether or not a LiDAR observation falls within a given node's range, indicating that the ray reflected off of an obstacle or has reached its maximum distance. The nodes are iteratively updated with each incoming LiDAR datum and eventually forms a complete high-resolution map of the robot's surroundings. Given these frameworks' octree-based structure, their fast update speed and low barrier to entry, they have become a popular choice for many roboticists' real-time mapping tasks. However, while beneficial for cluttered indoor and urban environments, OctoMap can leave significant areas in its terrain estimate unfilled, particularly at longer distances where hit densities are lower. This makes OctoMap and its derivative systems unreliable for the task of surface reconstruction in an outdoor, off-road context.

Machine learning models, such as SVMs, have been used to infer surface points at unobserved locations with reasonable success [11]. However, the actual terrain reconstruction and estimation process takes place offline. The GPMap [12] and GPOctoMap [13] frameworks are both terrain mapping and surface estimation approaches that use input data from the LiDAR to train a set of Gaussian Process (GP) regressors which fill in any gaps in the surface estimate. Both also use octree-based maps to discretise the environment and make the terrain data more amenable to additional processing. However, GPs are well known to be computationally expensive and, despite optimisations introduced by various authors to improve their scalability [14, 15], they remain as such, making GPMap and GPOctomap unsuitable for real-time mapping and surface reconstruction tasks for reasonably-sized environments.

Other methods have been proposed for efficiently reconstructing the complete surface of an arbitrary 3D object [16]. One approach is to model the surface using an energy function and optimise it according to some structural parameters [17, 18]. Such functions are relatively straightforward to minimise and present an efficient surface reconstruction pipeline. However these techniques have only been applied to 3D objects' surfaces and have not been used for environmental mapping. Indeed, given the flexibility of these methods, their proven computational efficiency and the problem at hand, we believe that an energy

model-based approach could prove useful for terrain reconstruction tasks in an outdoor off-road setting.

3 Method

At their cores, mapping an outdoor off-road terrain and an urban one are functionally similar processes. OctoMap [1] and UFOMap [2] are popular, robust and freely available tools that can produce occupancy maps of various environments. However, they do not have the capability of inferring whether a node in an unknown occupancy state between other occupied nodes is occupied or free. This is an important factor for producing complete and accurate surveyance maps, as well as robots that require complete knowledge of the surrounding terrain to plan their movement and accomplish their tasks more effectively. The fact that rural outdoor environments are often continuous, albeit uneven, surfaces can be exploited to fill in these gaps and estimate a node's state if it is located on the same contour as its occupied neighbours.

Our approach, called EMap (Energy Minimisation Mapping), uses the aforementioned assumption of a continuous surface contour to find the surface estimate that best fits the input LiDAR reference data. As the name alludes to, EMap is based on the concept of energy minimisation (EM, sometimes called 'geometry optimisation'), which is a process to determine the optimal geo-spatial arrangement of objects according to an energy-based model. This approach is often used in computational chemistry to find the expected geometric arrangement of atoms in a molecule based on their inter-atomic energy bonds (a spring-like force that attracts or repels a pair of atoms based on the distance between them). The atoms' final resting positions will be located where their inter-atomic forces settle at a new equilibrium state and the net energy in the system is zero.

Applying this approach to a LiDAR-based terrain mapper is fairly straightforward. Suppose that at time t we have a set of surface nodes to approximate the terrain surface. Like the atoms discussed earlier, these nodes are connected to one another according to an energy model that is a function of the relative distance between neighbouring nodes (i.e. the spring-like forces described earlier). Incoming LiDAR point data at time $t + 1$ are then modelled as new nodes with additional spring forces between a new node and its nearest neighbour from the surface nodes from time t. This approximates additional energy being introduced to an enclosed system, resembling work done on the surface node system. The surface nodes are forced absorb the incoming energy and reach a new equilibrium by adjusting their relative positions. Since we are only interested in the map's surface topology, we can simplify the problem by fixing the surface nodes' x and y coordinates in the LiDAR's local frame, limiting all the nodes' displacements to the z dimension only. Refer to Fig. 3 for an example spring-node system exposed to an incoming LiDAR point.

Let us now mathematically formalise our EMap approach. The terrain surface estimate consists of a set of nodes with constant (x, y) coordinates and variable heights, z, measured relative to the LiDAR sensor's local frame. These

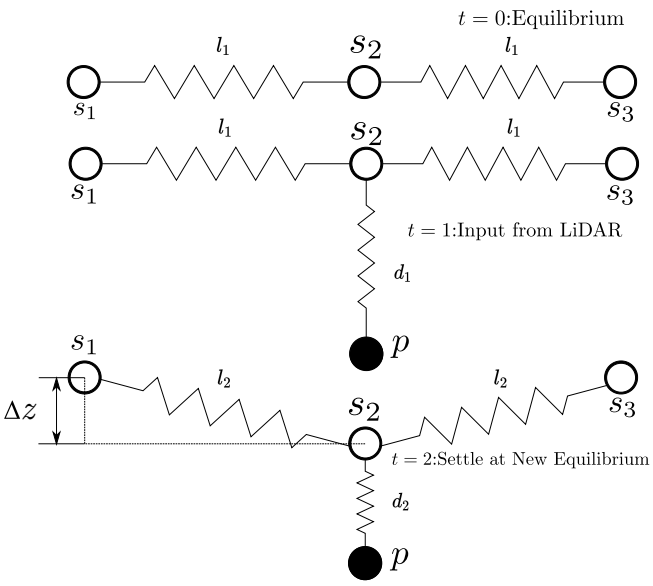

Fig. 3: Interactions between between nodes, neighbours and LiDAR points.

are attached to one another in a grid pattern with springs with a spring constant k. A node s will therefore have between 2 and 4 neighbouring nodes influencing s's position, depending on its position in the grid (e.g. a corner node will only have 2 connected neighbours). The total energy at a single node s at equilibrium at timestep $t = 0$ is then given by

$$E_s^{t=0} = \sum_{i=0}^{n} \frac{1}{2} k \Delta z_{si}^2, \tag{1}$$

where n is the number of connected neighbours s has and Δz_{si} is the vertical-only difference in height between nodes s and i (since x and y are fixed, their deltas are eliminated from the spring displacement vector).

Incoming data from the LiDAR are modelled as work done on the surface estimate nodes, perturbing them from their equilibrium states. The work done by each incoming LiDAR point is modelled as another spring connecting each input point with its nearest surface node. In its equilibrium state at $t = 0$ and $\Delta \boldsymbol{z} = 0$, the total net energy in the surface estimate's spring network is zero. However, when new LiDAR points are introduced at timestep $t = 1$, the system's net energy is no longer zero and the resultant energy at each node is calculated as

$$E_s^{t=1} = \sum_{i=0}^{n} \frac{1}{2} k_i \Delta z_{si}^2 + \sum_{j=0}^{m} \frac{1}{2} k_j d_{sj}^2, \tag{2}$$

where m are the number of input LiDAR points attached to node s and d_{si} is the Euclidean distance between point j and node s. This perturbation results in a non-zero energy state for the surface estimate and $\Delta \boldsymbol{z}$ must be adjusted to compensate for the new energy introduced to the system. We ignore the springs' transient behavior in favour of the steady-state response and solve the system as a set of linear equations to determine $\Delta \boldsymbol{z}$. Rewriting Eq. 2 in the form $A\boldsymbol{x}+b=c$ gives,

$$\boldsymbol{e} = \frac{1}{2} K_s \Delta \boldsymbol{z} \ + \frac{1}{2} K_d \boldsymbol{d}, \tag{3}$$

where $\boldsymbol{e}$ is an $a \times 1$ vector containing the energy E_s at each node, a is the number of nodes in the surface estimate and K_s is an $a \times a$ diagonal matrix containing each node's effective spring constant (e.g. a node with 4 neighbours and $k = 10\mathrm{N\,m^{-1}}$ will have an effective $k_{eff} = 40\mathrm{N\,m^{-1}}$). $\Delta \boldsymbol{z}$ is an $a \times 1$ vector made up of the sum of the squared spring displacements, Δz_{si}^2, between the surface node and its neighbours and $\boldsymbol{d}$ is an $a \times 1$ vector containing the sum of the squared displacements between node s and its nearest m input points, d_{sj}^2 (see Eq. 2). Note that the term $\frac{1}{2} K_d \boldsymbol{d}$ in Eq. 3 represents a fixed quantity of work that is done on the system by the input LiDAR points, and therefore remains unchanged. By setting $\boldsymbol{e} = \vec{0}$, we can solve Eq. 3 analytically solved at the new equilibrium state. The required displacement that must be applied to each surface node to reach this new equilibrium state is determined by taking an element-wise square root of $\Delta \boldsymbol{z}$. This process of performing work on the surface nodes and finding their new equilibrium positions is iterative and takes place for each LiDAR scan, refining the overall terrain estimate over time.

To find each node s's nearest input LiDAR points, we use a kD-tree search method, which is $\mathcal{O}(\log a)$ in complexity. However, the overall complexity is dominated by the matrix inversion process to solve Eq. 3. With our current constract of fixing the nodes' (x, y) coordinates, K_s is diagonal and can be inverted in linear time. However, we might relax this constraint in future work and we therefore cannot rely on the aforementioned diagonality to remain true indefinitely. Were therefore opt to use a conjugate gradient descent method to remain flexible and determine K_s^{-1} (K_s will always be semi-positive definite). This results in a complexity of $\mathcal{O}(a\sqrt{\kappa})$, where κ is K_s's condition number which is expected to be small. We can therefore anticipate a time complexity that grows linearly with the number of nodes in EMap's surface estimate.

4 Experiments

4.1 Setup

A Gazebo simulation was created to allow a LiDAR-mounted vehicle to drive over a surface at $1\,\mathrm{m\,s^{-1}}$ and collect point data. The simulation environment geometry, shown in Fig. 1, is such that the angular pose and position of the LiDAR remained unchanged, despite variations in the surface shape the vehicle

was driving over. This mimics the behaviour of a real large agricultural vehicle or UAV, neither of which are significantly affected by variations of their work surfaces (UAVs can keep their altitude quite stable in favourable weather conditions). Furthermore, the robot was limited to driving only forward along the x-axis and no steering or planning was implemented. The LiDAR sensor simulated is an Ouster OS1-64 with 512×64 beams in a 360° view around the sensor and 22.5° above and below its horizontal. Sensor noise was added and is simulated as ($\pm$0.05m) Gaussian noise. Only points within a 10 m radius from the sensor are considered. The terrain surface was generated with $z_s(x_s, y_s) = h(x_s, y_s) = \sin \frac{x_s}{\pi} \sin \frac{y_s}{\pi}$. The surface's sinusoidal pattern of peaks and troughs introduce periodic occlusions, thereby guaranteeing gaps in the sensor data and providing a challenging mapping task. In order to determine a realistic surface estimation error, we repeated the experiment 10 times, setting the robot's initial position along the y-axis at a new value for each run, 1m apart. This provides sufficient resolution across the entire surface profile. The collected LiDAR data were processed by both the EMap and OctoMap systems.

EMap Our EM-based method begins by transforming the incoming LiDAR data to the vehicle's local frame and filtering all points that fall outside the 10m range. A kD-tree nearest neighbour search is then used to find the Euclidian distances between the incoming LiDAR points and their nearest surface nodes, followed by the EM process that determines the surface nodes' required displacements, which are then added to their z coordinates. To cover the 10m work surface, the surface estimate was set to contain 40 nodes in the vehicle's x and y directions, giving 1600 surface nodes in total.

OctoMap As a baseline, we implemented a simple interpolation layer on top of OctoMap to produce surface estimates. OctoMap was used rather than UFOMap, as it is mature and widely-used, and the authors of UFOMap did not report any significant increases in accuracy over OctoMap. OctoMap was used to build up a volumetric occupancy map with the incoming LiDAR data as it normally does. However, since the work surface is flat and continuous, we know that the topmost occupation nodes will form the surface estimate. Another 40×40-node surface estimate is then set to assume the position of these topmost octree nodes and model the underlying surface. The additional computation takes the form of a kD-tree search to find the surface's top layer and transform the surface nodes to that location.

4.2 Experiment Scenarios

We devised a number of example terrain mapping scenarios to evaluate the terrain mapping methods with. Each of these scenarios have their own set of constraints and priorities, reflected by a set of utility functions, which, for example, prioritise nearby points over those further away for obstacle detection purposes. These are two basic scenarios with minimal filtering, as well as two

drivability ones. The utility functions, F, are applied to the surface estimates like a filter and range with $F(x, y) \in [0, 1]$. The mean square error (MSE) is,

$$\text{MSE} = \sum_t \frac{1}{a} \sum_a ((g(x, y)_t - h(x, y)_t)F(x, y))^2, \tag{4}$$

where g and h are the surface estimates given by EMap or OctoMap, and the ground truth, and a the total number of nodes in the surface estimate.

Baseline - Raw Points This scenario is a naive scan including all input points equally, with a utility function $F_r(x, y) = 1$, and acts as the baseline scenario.

Mapping - Nearest Points A nearest point scan simulates cases where only the points closest to the sensor are considered reliable. This gives the utility function

$$F_n(x, y) = \begin{cases} 1, & \sqrt{x^2 + y^2} < 0.25\text{m} \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Drive Planning - Gaussian The first drivability scenario emphasises points at a middle distance, r_g, from the vehicle, while progressively ignoring points further away from r_g, mimicking the needs of a planner for autonomous vehicles:

$$F_g(x, y) = \exp\Big(-\frac{1}{2}(\sqrt{x^2 + y^2} + r_g)^2 \Big). \tag{6}$$

Obstacle Detection - Cumulative Distribution Function This second drivability scenario prioritises all data points closest to the sensor, which decays with the points' distance to the sensor and allows for effective obstacle detection nearby a vehicle. This utility function can be modelled by an un-normalised cumulative distribution function (CDF) centred around a threshold scanner range, r_s,

$$F_c(x, y) = 1 - \frac{1}{2}\Big(1 + \frac{\text{erf}(\sqrt{y^2 + x^2} - r_s)}{\sqrt{2}}\Big). \tag{7}$$

The values for r_s and r_g were heuristically set to 4 m and 6 m.

5 Results

5.1 Reconstruction Accuracy

The MSE results for all of the scenarios across the 10 experiment runs are given in Table 1. These show EMap consistently generating more accurate terrain estimates for all of the experiment scenarios, improving upon the OctoMap baseline

between approximately 29% and 44%. The standard deviations for the MSE's are also significantly reduced for the EMap system, indicating that the estimates are more precise in addition to being more accurate. Overall reductions in MSE in each scenario show that the effective scanning range, r_m, is extended when using EMap for terrain mapping. For example, the scanning range for the *CDF* scenario can be increased by 36.7%, to 5.47 m, over the baseline's 4 m without suffering a decrease in accuracy compared with OctoMap. This conclusion is further supported by the surface estimates' error spreads from the *raw* scenario shown in Fig. 6. These heatmaps show OctoMap's MSE significantly spiking around the edges furthest away from the sensor along the x-axis. The opposite is observed from EMap's result – the MSE is lowest furthest way from the sensor on the x-axis. The error scales for these two sets of results are quite different, so we consider the normalised MSEs. These further reinforce EMap's error values being smaller in general and also more consistent compared to OctoMap's results.

Table 1: Each scenario's terrain reconstruction accuracy for OctoMap and EMap.

Scenario	OctoMap	EMap	% Difference
Raw	74.6 ± 18.2	45.8 ± 6.3	38.6%
Nearest Points	88.8 ± 51.4	50.0 ± 25.6	43.7%
Gaussian	1.21 ± 1.19	0.86 ± 0.67	28.9%
CDF	0.93 ± 0.58	0.59 ± 0.32	36.6%

The absolute MSE values in isolation are quite large, e.g. 45 cm for EMap's *raw* scenario. However, this can be explained by the cyclical, sinusoidal terrain that was used for the experiment, where the error fluctuates periodically with the vehicle's forward movement as the terrain transitions from a peak to a trough (see Fig. 5). This is because a peak occludes a portion of the terrain from the LiDAR's view, with its view only fully restored when the vehicle crests the peak and sees behind it.

5.2 Processing Time per Loop

Fig. 6 shows a plot of EMap's processing time per loop as a function of a, the number of nodes in the surface estimate. The time was taken as the mean time per loop across a 30s period for multiple values of a. Considering its formulation, EMap's computational complexity is expected to be linear and dependant on a only (see Sec. 4.1). Indeed, this expectation is confirmed by Fig. 6, where the processing time per loop grows linearly with the number of surface nodes and indicates that EMap can be scales reasonably well – doubling the surface resolution increases the computation time by a relatively low 25%. In the experiments, 1600 nodes were used, giving 16 nodes per m^2 and taking approximately 0.05 s

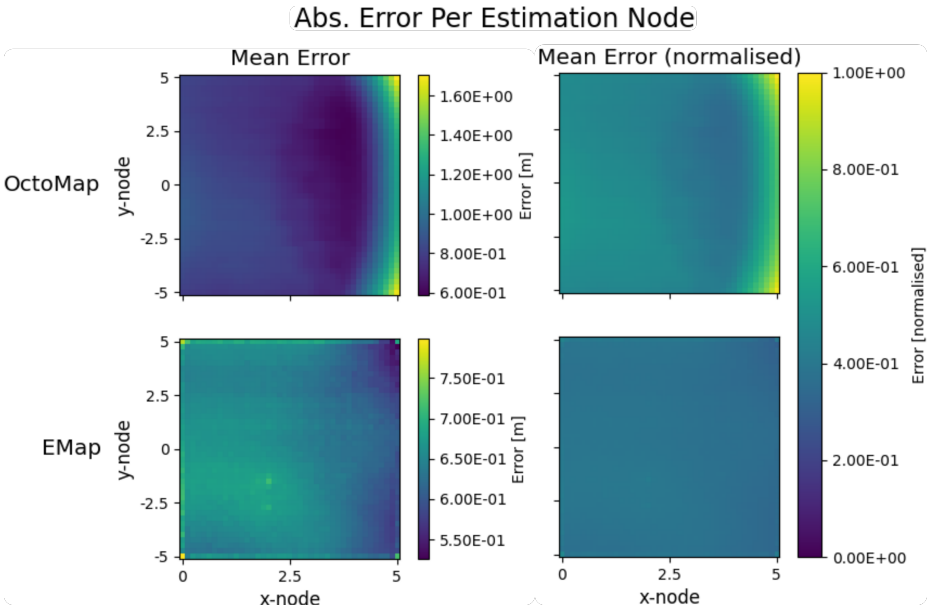

Fig. 4: Histogram of the MSEs recorded for both EMap and OctoMap during the Raw scneario at each node on the estimation surface.

to process, giving a 20 Hz update rate, achieving our goal of real-time terrain reconstruction.

6 Discussion

EMap showed a consistent reduction in MSE compared to the OctoMap baseline with a reasonable computational cost, allowing the terrain reconstruction process to be run in real-time. Beyond the improved accuracy over the baseline, the extension to r_m facilitated by EMap will lead to a direct reduction in the number of scans needed to build an accurate map. When this reduction is applied to an industrial-scale terrain mapping operation, e.g. large autonomous agricultural machinery, it could be very beneficial in reducing the overall operational cost and environmental impact.

The simulation experiments carried out in this work are sufficient for proving the viability of EMap as a concept. However, there are numerous limitations from using a sinusoidal terrain and simulations and additional work within more complex simulated and real environments are needed to properly test EMap's viability as a terrain mapping solution. Nevertheless, the results from this proof-of-concept work is promising and indicates that EMap is an avenue worth investigating more.

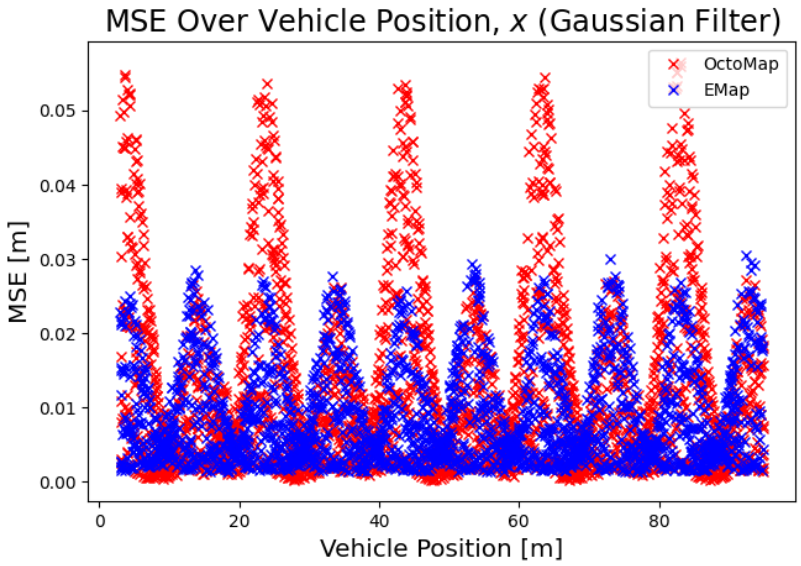

Fig. 5: The MSE from for over the distance the vehicle travelled during the Gaussian scenario as determined by EMap.

7 Conclusion

Based on results generated from simulation experiments, EMap reduces the MSE by up to 43.7% over Octomap, and extends the effective scanning range by 25% for one of the mapping scenarios, compared to that of the baseline. This added benefit comes at little additional computational cost and is accomplished in real time – though higher-resolution terrain maps can also be generated offline. For robotic mapping tasks, this could lead to significant cost savings and allow operators to generate more reliable long-distance traversal plans, further improving operational efficiency.

Future work should look into applying the EMap approach to a real-world scenario to determine its effectiveness therein. Furthermore, relaxing the strict condition of locking the nodes' (x, y) coordinates can be investigated to determine whether it can further improve EMap's surface estimation capabilities.

References

1. A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.
2. D. Duberg and P. Jensfelt, "Ufomap: An efficient probabilistic 3d mapping framework that embraces the unknown," *arXiv preprint arXiv:2003.04749*, 2020.

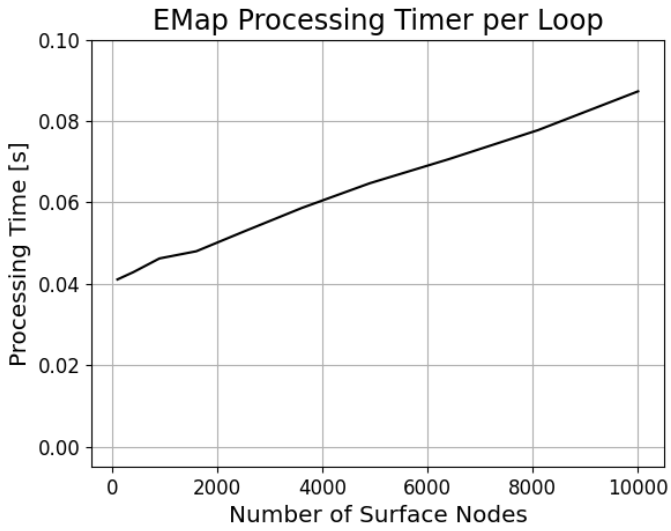

Fig. 6: The mean time taken by EMap to fit a surface to a new input cloud per LiDAR update loop.

3. D. Gingras, T. Lamarche, J.-L. Bedwani, and É. Dupuis, "Rough terrain reconstruction for rover motion planning," in *2010 Canadian Conference on Computer and Robot Vision*. IEEE, 2010, pp. 191–198.
4. F. Malartre, T. Feraud, C. Debain, and R. Chapuis, "Digital elevation map estimation by vision-lidar fusion," in *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2009, pp. 523–528.
5. J. Sock, J. Kim, J. Min, and K. Kwak, "Probabilistic traversability map generation using 3d-lidar and camera," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 5631–5637.
6. K.-W. Chiang, G.-J. Tsai, Y.-H. Li, and N. El-Sheimy, "Development of lidar-based uav system for environment reconstruction," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 10, pp. 1790–1794, 2017.
7. U. G. Survey, "The national map—new data delivery homepage, advanced viewer, lidar visualization," US Geological Survey, Tech. Rep., 2019.
8. C. Hladik and M. Alber, "Accuracy assessment and correction," *Remote Sensing of Environment*, vol. 121, pp. 224–235, 2012.
9. I. Werbrouck, M. Antrop, V. Van Eetvelde, C. Stal, P. De Maeyer, M. Bats, J. Bourgeois, M. Court-Picon, P. Crombé, J. De Reu *et al.*, "Digital elevation model generation for historical landscape analysis based on lidar data, a case study in flanders (belgium)," *Expert Systems with Applications*, vol. 38, no. 7, pp. 8178–8185, 2011.
10. D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.
11. C.-W. Yeu, M.-H. Lim, G.-B. Huang, A. Agarwal, and Y.-S. Ong, "A new machine learning paradigm for terrain reconstruction," *IEEE Geoscience and Remote Sensing Letters*, vol. 3, no. 3, pp. 382–386, 2006.

12. S. Kim and J. Kim, "Gpmap: A unified framework for robotic mapping," in *Field and service robotics*. Springer, 2015, pp. 319–332.
13. J. Wang and B. Englot, "Fast, accurate gaussian process occupancy maps via test-data octrees and nested bayesian fusion," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1003–1010.
14. D. Eriksson, K. Dong, E. Lee, D. Bindel, and A. G. Wilson, "Scaling gaussian process regression with derivatives," in *Advances in Neural Information Processing Systems*, 2018, pp. 6867–6877.
15. V. Tresp, "A bayesian committee machine," *Neural computation*, vol. 12, no. 11, pp. 2719–2741, 2000.
16. M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva, "State of the art in surface reconstruction from point clouds," in *Eurographics 2014 - State of the Art Reports*, 2014.
17. C. Leung, B. Appleton, B. C. Lovell, and C. Sun, "An energy minimisation approach to stereo-temporal dense reconstruction," in *Proc. ICPR*, 2004.
18. P. Labatut, J.-P. Pons, and R. Keriven, "Robust and efficient surface reconstruction from range data," in *Computer graphics forum*, vol. 28, no. 8, 2009, pp. 2275–2290.