Journal of the Brazilian Computer Society, 2024, 30:1, doi: 10.5753/jbcs.2024.3795

Global Localization using OpenStreetMap and Elevation Offsets

André Przewodowski [**Institute of Mathematics and Computer Sciences, Universidade de São Paulo** | ***carlos.andre.filho@usp.br***]
Fernando Santos Osório [**Institute of Mathematics and Computer Sciences, Universidade de São Paulo** | ***fosorio@usp.br***]
Valdir Grassi Junior [**São Carlos School of Engineering, Universidade de São Paulo** | ***vgrassi@usp.br***]

André Przewodowski and Dr. Fernando Osório are with the Institute of Mathematics and Computer Sciences. Dr. Valdir Grassi Jr. is with the São Carlos School of Engineering. Both institutes are from the Universidade de São Paulo, Av. Trab. São Carlense, 400, Centro, São Carlos, SP, 13566-590, Brazil.

Received:29 October 2023 • **Accepted:** 08 May 2024 • **Published:** 15 September 2024

Abstract Localization is a critical component in autonomous vehicle navigation stacks. While GNSS-only localization cannot be fully reliable and available all the time, localization based on 3D high-definition (HD) maps have to be robust to world changes, which is still a challenging issue. Added to that, in general, HD maps are expensive and difficult to construct and maintain. In this paper, we propose a particle filter-based 2D global pose estimation method that can use the crowdsourced OpenStreetMap (OSM) API, a digital surface map, or both. The main contributions of the proposed approach are: that it is lightweight, does not require the vehicle to map the environment, does not require a GPU (can be used with low-power computing resources), is agnostic to the odometry source, and achieved relatively low position and orientation errors for this localization modality using the KITTI dataset sequences. The proposed method's implementation is open source and is available with the experimental results on our GitHub page.

Keywords: Autonomous driving, data association, global localization, self-driving vehicles

1 Introduction

In the autonomous driving industry, localization plays an important role in navigation systems. One example is that motion and trajectory planning both depend on the knowledge of the agent's pose.

Localization is generally estimated using a filtering approach (backend) that fuses local and global pose estimations. Local pose estimation approaches provide the vehicle's pose with respect to a relative frame - usually placed at the sensor's origin. They provide pose estimates at a high frequency (generally above 10Hz) but are susceptible to drift in the long run. Drifting is the accumulation of errors in the process of estimating relative poses and is mitigated by applying place recognition and loop closure techniques using local maps constructed along the trajectory.

On the other hand, global pose estimation techniques provide the agent's position and orientation with respect to a static and known reference frame. One example is the Universal Transverse Mercator (UTM) zone origin on which the vehicle is placed. Global poses can be retrieved using place recognition on maps constructed on a previous drive using offline simultaneous localization and mapping (SLAM) approaches or GNSS-based correction.

GNSS-based localization is not reliable on urban canyons[1]. Also, dense foliage around the vehicle impacts the accuracy of the measurements. As an alternative for improving the estimations, expensive GNSS devices are fused with inertial measurement unit (IMU) devices for improving localization. Even in that case, some devices can interfere with or intentionally block GNSS signals (GNSS jammers and special security/military devices). Therefore, localization pipelines usually use more sources of global localization, since the first two problems still occur.

Map-based localization using high-precision 3D maps requires reconstruction using one or more light detection and ranging (LiDAR) devices, which are expensive. Among the issues of the reconstruction process, filtering dynamic objects, performing manual point cloud alignment, ensuring very precise local estimations and the volume of the reconstructed maps are problems that constantly occur. Furthermore, when performing localization using those maps the vehicles also need to use the LiDAR (with proper intensity calibration), which does not help in reducing the project's price, being restricted to expensive customized cars.

Other types of maps that can be used as an alternative to high-precision 3D maps can also be used. For instance, digital elevation models (DEM) and digital surface maps (DSM) represent the surface's elevation and are usually recorded using LiDAR-equipped unmanned aerial vehicles. The DEM represents the ground plane elevation profile and disregards trees and buildings, while the DSM represents the elevation profile considering trees, buildings, and other elements that do not belong to the ground plane. While very popular in the agriculture context, they are used in Geographic Information Systems (GIS) and can be a valuable tool in the task of global

[1]Regions with tall buildings in the surroundings that interfere with the satellites' signal.

localization in the autonomous vehicles' context. In addition, they are much less expensive to reconstruct than 3D HD maps. Furthermore, some countries have a very broad DSM and DEM coverage conveniently gathered by open source tools, for instance, the Equator studios tool[2].

Another type of map is the OpenStreetMap[3] (OSM), a crowdsourced map containing the road network, places of interest, or even high-level-represented landmarks that can be used to aid in the visual place recognition task. The advantage of this type of map is that it is a free source and there are many available tools that make its usage easier, as the OSMnx[4], Pyrosm[5] and JOSM[6].

In this paper, we use the benefits of the OSM and the DSM maps for estimating global poses. We propose two approaches that can be used either individually or together: the "Global Localization using the OSM" (GLOSM) approach, which uses the OSM road network, and the "Elevation Offset" (ELOFF)-based approach, which uses the DSM for corrections. In short, the core concept is to use the navigable area, provided by the OSM, and the elevation, provided by the DSM, for weighing a particle filter. The output is a global pose that can be used as input for a state estimation backend, for instance, an Extended Kalman Filter or a pose graph.

The proposed approach was tested using the KITTI dataset sequences from 0 to 10 and it could achieve translational error metrics close to $3m$ and orientation error lower than 1^o. It was tested and compared using three different odometry sources (from different sensor modalities: camera and LiDAR) and it was able to reduce the drift errors to less than 10%.

The main contributions of the proposed methods are:

- They can use open and crowdsourced data without requiring expensive offline processing or mapping steps;
- They are robust to GNSS failures and do not depend on 3D HD maps;
- They can be used on machines with lower computational power so that they can be incorporated into conventional vehicles at a lower cost.

Additionally, we also provide:

- A LiDAR odometry implementation for testing the localization algorithms (named SLOPY);
- Tools for constructing a digital surface map from laser scans collected by a ground vehicle.

The source code of the global localization methods, tools, and experimental results are available in our GitHub repositories[7][8][9].

[2]Available at: https://equatorstudios.com/
[3]Available at https://openstreetmap.org/
[4]Avaiable at https://osmnx.readthedocs.io/
[5]Available at: https://pyrosm.readthedocs.io/
[6]Available at https://josm.openstreetmap.de/
[7]SLOPY: github.com/cabraile/SLOPY
[8]Map construction tools: github.com/cabraile/3D-Kitti-Mapper
[9]GLOSM, ELOFF, and results: github.com/cabraile/GLOSM-ElOff

2 Related Work

In this paper, we explore different aspects of the localization problem, which is related to other works that use OSM, elevation profiles, aerial imagery, semantic maps, and 2D geometric features for localization. In this section, we provide a broad overview of those related methods.

2.1 OSM-based localization

The OSM-based localization approaches tend to use higher-level world representations such as text, road networks, or the existence of buildings in the surroundings in order to perform data association. The process generally requires extracting those higher-level data from the low-level data of the sensor information for matching with the map data.

In Radwan *et al.* [2016], the authors perform data association between the text detected from a camera image and the text contained in the OSM nodes in the surroundings, which were filtered using received GNSS coordinates. Their method does not use sequential information in order to improve the estimated global localization and is highly dependent on the GNSS. In contrast, our method, as can be seen in Section 3, depends on the GNSS only for initializing the global localization. Our approach does not require text detection and uses sequential information in order to provide global localization.

In Brubaker *et al.* [2016], they propose a new state representation model based on the displacement and orientation with respect to a road segment (way) and model the global state distribution using a Gaussian Mixture Model. Their idea is to predict the state based on the vehicle's displacement velocity and the road bifurcations ahead while updating the hypotheses' weights using visual odometry. Their work inspired our GLOSM in using the road segments as constraints for the localization problem. However, we model differently the way those constraints are used, so that we work directly with the cartesian state variables instead of working with an intermediate representation.

The method presented in Yan *et al.* [2019] performs global localization using 4-bit descriptors based on building gaps and road intersections retrieved from the OSM API. Even though compact, building this descriptor requires using semantic range images. Also, walls and fences that surround buildings are not always mapped in OSM, which frequently occurs in residential neighbors, and is not handled in their approach. Our GLOSM approach, on the other hand, is agnostic to the sensor used and is not prone to issues related to unmapped walls.

In Amini *et al.* [2019], an end-to-end approach deep learning stack for navigation was proposed, in which the camera images, added to GNSS coordinates and the OSM map of the surroundings' driveable area, are received as input and the steering control and refined localization estimation are provided as output. More recently Sarlin *et al.* [2023] proposed the OrienterNet: a global 2D pose estimation method that uses the coarse pose, monocular camera images and the rasterized OSM geometric and semantic information for estimating the probability distribution of the camera in the given map using convolutional neural networks. The network extracts the semantic BEV from the camera image and encodes the OSM shapes in order to perform exhaustive BEV map matching. While our GLOSM approach can use GNSS to complement the filter process, GNSS is not required for localization which is a benefit compared to the approach proposed by Amini *et al.* [2019]. Added to that, neither GLOSM nor ELOFF uses deep learning - which requires dedicated hardware, heavy offline processing (training) and an extensive dataset that covers most of the possible cases for good generalization.

2.2 Elevation-based localization

Elevation does not tend to be a unique feature in elevation-based maps, since many places in a map can have the same elevation. Still, when fused with the appropriate filtering methods, it can be used for reliable global pose estimation. In Mandel and Laue [2010], the authors propose using the elevation computed using the pressure measured by a barometer for correcting the position of an autonomous wheelchair. Similar to our approach, the position of the particles is corrected by comparing the deviation between the elevation and a digital elevation model (DEM). Their approach, however, depends on the barometer for computing the elevation - which is prone to pressure changes in the atmosphere and constantly has to be recalibrated. We, instead, propose using the odometry displacement with respect to the fixed world frame's upward axis.

On a different application, Hyuga *et al.* [2016] proposed using a smartphone's barometer information for filtering hypotheses of subway stations a passenger can be. In their approach, they estimated the altitude above sea level and found the matching stations according to the prior altitude map of each available station and with the constraints of which line of the subway network the user was on.

In Larnaout *et al.* [2013], both GNSS and DEM are used for global 3D pose estimation using monocular SLAM. In their approach, the DEM is used for adding a constraint in the pose graph, which uses the reprojection error of the road plane points as part of the optimization's cost function.

In Imperoli *et al.* [2018], a more complex system that integrates different sources of odometry and global position was proposed in the context of unmanned farming ground vehicles. In their application, the DEM was used as a pose graph constraint in the fixed world frame's upward axis in order to avoid elevation drift. Our proposed method ELOFF uses the DSM as a weighing constraint in the x-y plane for the particle filter.

2.3 Geometry-based localization approaches

While not exactly a higher-level feature, the available information on buildings' 2D footprints is also interesting and can also be used for performing data association and directly correcting the estimated vehicle's pose. The work presented in Bhattacharyya *et al.* [2017] uses both the 2D building footprints and the DSM for retrieving 3D information on buildings in Tokyo. With the 3D buildings, the 3D point clouds from stereo images are used for correcting the 2D pose estimated using an inertial navigation system. Similarly, in Quack *et al.* [2019], their localization correction relies on matching laser scans with 2D building polygons. In their approach, those building polygons were annotated manually on their test site using satellite imagery. The scan matches are used for correcting the Monte Carlo Localization (MCL) filter. The backend of our localization pipeline used for integrating the GLOSM and ELOFF estimations is the MCL as in the approaches mentioned, but we do not use the map's geometric information for performing data association or for directly correcting the poses.

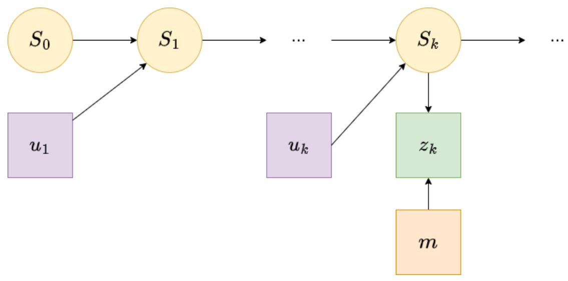


Figure 1. The Bayes network that models the conditional independency relationships for our approach. Notice that the element m does not contain a subscript, which is intended since the map m is assumed to be static. In the context of this model, lowercase variables represent observed quantities and uppercase variables represent variables that cannot be directly measured.

While not explicitly using the map geometries for correcting the vehicle position, in Miller *et al.* [2021] they perform global localization by matching semantically annotated scan point clouds with the semantic-segmented aerial imagery of the world. The core idea is that the segmented scans can be projected for each particle and compared to the semantic map of that particle's position. This approach requires: training a semantic segmentation model for segmenting aerial images; segmenting the map's aerial images, which can take a long time to cover small regions even using a lower resolution; and performing on-the-fly point cloud semantic segmentation, which adds a computational burden to the pipeline.

3 Method

The goal of the proposed method is to correct the drift from an odometry source using global position estimates from a given digital surface model (DSM) map or from the OpenStreetMap layers. Hence, the state to be estimated is composed of a 3D position and the yaw orientation ψ, thus it is represented by $S_t = (x_t, y_t, z_t, \psi_t)$. The subscript t denotes the time stamp in which the state is contained. Differently from the SLAM problems, the map m components are not included in the state space since the map is assumed to be given and static. Considering this, we decided to model the conditional relationship between the variables as in the Bayes network illustrated in Figure 1.

The probability distributions of the estimations using the GLOSM or the ELOFF approaches are not expected to be Gaussian and neither the map measurements are expected to be obtained using linear transformations. Also, the model represented by the presented Bayes network is conveniently easy to use if implemented using the particle filter framework, which was adopted as our sequential filtering approach.

If we assume that we have an odometry source (obtained using any arbitrary sensor) - which provides relative poses u_t between time steps $t-1$ to t -, and if we know the distribution from which u was sampled, we only need to define the probability distribution of the measurement model $P(Z|S,M)$, where Z represents the measurement (and z would represent the observed quantity), S the state vector and M the map source used, which can be the OSM, $M = m_{OSM}$, or the digital surface map, $M = m_{DSM}$. In this paper, we compared two odometry sources: SLOPY (ours, see Section 3.3) and DF-VO [Zhan *et al.*, 2021, 2020], but the filter implementation is modular enough for using other odometry sources as well.

In general, the implemented particle filter works as follows:

1. The pose offset estimated from the odometry algorithm is provided as an input for the particle filter's sampling step;
2. The particles are weighed using the OSM (GLOSM) or the DSM (ELOFF) models;
3. The weighed particles are then resampled.

As for the first step (known as the sampling step), it is performed as in any particle filter approach: from the estimated pose offset, assumed to have a known noise model[10], each sampled particle is displaced with noise.

The second step consists of measuring the likelihood of the hypothesis each particle represents so that particles that have more likely states have higher weights. In this paper, this is done by comparing the particles' positions with the map. Depending on the map used this is done differently, as described in Section 3.1 (GLOSM) and in Section 3.2 (ELOFF). We emphasize that, in most global localization approaches, this is performed with data association between features detected by one of the sensors and a map. In our approach, the key concept is to use the trajectory to weigh the particles instead of performing feature matching.

Finally, resampling the particles will keep the most likely particles and discard most of the less likely particles. In our implementation, we used the low-variance sampler approach, which is less prone to bias when generating random numbers.

3.1 GLOSM

The OpenStreetMap[11] is a rich resource, which consists of many higher-level landmarks, the road network geometry, and its semantics. In this paper, we only explored the navigable areas provided by the road network.

The road network provides valuable data for localization: the driveable area. Assuming the vehicle is in a driveable area or assuming that an external module is able to provide this information, we can weigh particles that are outside the driveable area. The Equation (1) represents the measurement model probability distribution when using this as a feature:

$$p(z = \text{+navigable}|s, M = m_{OSM}) \propto \begin{cases} \alpha & \text{if } s \text{ in a driveable area,} \\ 1-\alpha & \text{otherwise,} \end{cases} \tag{1}$$

[10]We assumed the noise to be Gaussian-distributed in our model

[11]At openstreetmap.org

where $0 < \alpha \leq 1$ represents how likely is the vehicle inside a navigable area.

Another feature provided by the OSM is the existence of traffic lights and stop signals inside a region. Even though their coordinates are not exact, they can be used to filter hypotheses by assigning low values to particles that are too far away for observing them when those features are detected. We explored this in our previous paper, Przewodowski and Osorio [2022]. However, those features are sparse in the dataset used for testing the proposed methods and we decided to not explore them further in this paper.

3.2 ELOFF

In this paper, we use an initial estimation of the absolute elevation of the vehicle and propagate the elevation offsets using the odometry estimates. Since the odometry estimation provides the relative pose from the current ego-vehicle frame (F_{curr}) to the same frame in the previous time step (F_{prev}), it is necessary to apply a frame rotation for retrieving the elevation offset with respect to the ground plane. This is represented in the following equation:

$$\Delta t_F^W = R_{F_{prev}}^W t_{F_{curr}}^{F_{prev}}, \tag{2}$$

where the rotation matrix R can be computed using the previous orientation provided by an IMU (already in the frame F) and $t_{F_{curr}}^{F_{prev}}$ represents the displacement from F_{prev} to F_{curr}, and Δt_F^W represents the displacement between the previous and current frame F had w.r.t. the world coordinates.

Having a digital surface map allows querying, for each particle, the expected elevation in its position. This is modeled in the following equation:

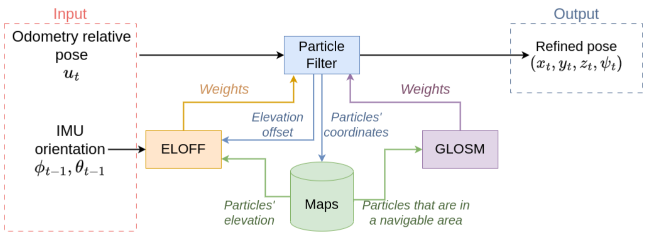
$$p(Z_{measured} = z|Z_{expected} = DSM(x,y)) \propto \frac{\sigma}{|(z - DSM(x,y))|}, \tag{3}$$

where z corresponds to the particle's estimated elevation after propagation using the odometry input, $DSM(x,y)$ retrieves the elevation in the (x,y) coordinates (in our case, projected in the UTM coordinates) and σ represents the uncertainty in the measurement (in meters). The right side of the equation is the inverse of the 1-D Mahalanobis distance which, in practice, will provide higher weights for particles that have an elevation closer to the map's elevation.

The operation for retrieving the elevation in the DSM raster given the global position of the object is known in geographic information systems (GIS). It consists of first converting from the projected coordinate frame to the image frame and then accessing the individual image value. The operation for converting from the global position to the image frame's coordinates is represented in the following equation:

$$\begin{bmatrix} x_I \\ y_I \\ 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & -S_x x_{min} \\ 0 & -Sy & S_y y_{max} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{4}$$

where S_x and S_y represent the map resolution (in pixels per meter), x_{min} and y_{max} are the minimum easting coordinate and the maximum northing coordinates of the map, and x_I and y_I correspond to the image coordinates column and row.


Input
Odometry relative pose u_t
IMU orientation ϕ_{t-1}, θ_{t-1}
Particle Filter
Output
Refined pose (x_t, y_t, z_t, ψ_t)
Weights
Elevation offset
Particles' coordinates
Weights
ELOFF
GLOSM
Maps
Particles' elevation
Particles that are in a navigable area
Figure 2. General data flow of the proposed method.

3.3 SLOPY

One of the inputs of our approach is the relative pose provided by an odometry source from any source, for instance: visual, inertial, or LiDAR. The estimated relative poses propagate the particles from the proposed particle filter. Even though we tested with different odometry sources (DF-VO and Viso2, which are camera-based), we also implemented an easy-to-integrate LiDAR odometry Python module named SLOPY (from "Simple Laser Odometry in PYthon") for conducting our experiments. While simple, it has demonstrated its effectiveness in combination with the proposed global localization approach.

SLOPY was implemented using a point-to-plane registration approach for computing the $SE(3)$ transformation matrix that relates the current frame with respect to the previous, T_t^{t-1}. Also, for making it more accurate, the linear and angular velocities are computed, so that the translation and rotational parts are separated from T_t^{t-1} and divided by the inverse of the scans' frequency. First, the Euler angles ϕ (roll), θ (pitch), and ψ (yaw) are obtained from the rotation matrix for computing the angular velocities. The velocities are then used for estimating an initial transformation, which improves the convergence of the ICP algorithm.

When a new scan is received in the time stamp $t+1$, the initial position for the point-to-plane registration is provided by computing the estimated displacement matrix $\tilde{T}_{t+1}^{t}$ assuming constant velocity and multiplying the velocities by the time interval between scans.

This framework allowed pose estimates to be provided every $100ms$ in the given pipeline. We emphasize that any other local estimation approach could be used, but using this approach was easier to prototype and was effective for our purposes.

3.4 Pipeline

The general pipeline data flow is illustrated in Figure 2. The received odometry poses, fused with a global orientation measurement (in this project we used the IMU as a source), are used for propagating the filter's particles. Then, using the particles' coordinates, we can retrieve the elevation of each particle and whether they are in a driveable area. The elevation information - along with the accumulated particles' elevation offsets - are used for computing the ELOFF weights (Equation (3)) and the information of particles contained in a driveable area is used for computing GLOSM weights (Equation (1)). The weights are then used for updating the particle filter, which provides the refined pose state (x_t, y_t, z_t, ψ_t).

As for filtering, the implementation of the proposed filter works the same as in usual Monte Carlo Localization (MCL) filters, in which the particles are sampled from a prior pose, then a cycle of sampling, weighing, and resampling starts.

In our approach, weighing the particles can be done using any of the measurement models proposed in Sections 3.1 (GLOSM) and 3.2 (ELOFF). We noticed that, in practice, weighing particles every period of time instead of doing it for every frame yields more stable results. In the case of ELOFF this might be related both to the accuracy of the digital surface map used and the low offset in the upwards direction between frames. In the case of GLOSM, usually, the lanes are not fully aligned with the true road positions or the map lanes are not as large as the true road. In that case, if we resample the particles on each frame good candidates might be filtered out as well. Instead, by weighing periodically, this effect is mitigated.

In Figure 3, one iteration of our pipeline's cycle is summarized. The resulting particles for each cycle represent the probability distribution of the trajectory given the odometry estimates and the measurements performed in the process. The moments of this distribution can be estimated empirically and provided as input for a localization pipeline, which can use factor graphs or any Kalman Filter variant approach as back-end processes.

4 Experiments

For validating our approach, the KITTI dataset [Geiger *et al.*, 2013], a well-known and open benchmark dataset used to evaluate and compare different intelligent and autonomous vehicles' approaches, was used. For comparing localization methods, the "odometry" subset was used[12], which is composed of 21 sequences (from "00" to "20") recorded under different environmental conditions, dates, and places. The sequences "00" to "10" are provided with the ground truth, while sequences from "11" to "20" do not have the ground truth publicly available (for a fair comparison between approaches). In its foundations, this dataset is targeted for comparing odometry and SLAM methods, not global localization, so that we could only evaluate trajectories "00" to "10" using the available references. The sequence "03" was not available for download by the time the proposed method was under its experimental phase, hence the experiments were not conducted in that sequence.

4.1 Building the DSM

Since the KITTI dataset does not provide a DSM and since the EquatorStudio tool's DSM for Karlsruhe has a very low resolution (30m), we reconstructed the environment using the LiDAR scans and georeferencing the point cloud using the inertial navigation system (INS) poses, which are also used as the ground truth. While in the mapping tools implemented (and mentioned in Section 1) the resolution could be as precise as the resolution of the used sensors, we chose to use a resolution of 15 centimeters per pixel, which allows a good tradeoff between map sizes, as the covered distances are big, and would allow frequent system updates if

[12]Available at: https://www.cvlibs.net/datasets/kitti/eval_odometry.php.

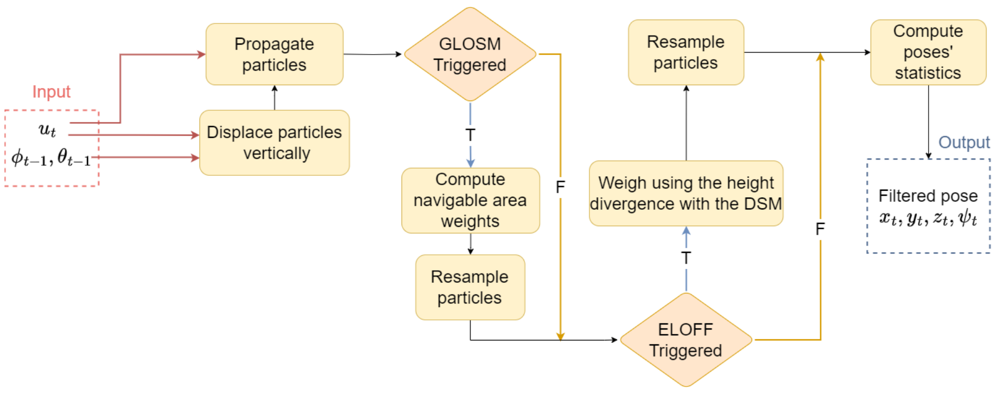

Input
u_t
ϕ_{t-1}, θ_{t-1}
Propagate particles
Displace particles vertically
GLOSM Triggered
T
F
Compute navigable area weights
Resample particles
ELOFF Triggered
Weigh using the height divergence with the DSM
Resample particles
Compute poses' statistics
Output
Filtered pose x_t, y_t, z_t, ψ_t

Figure 3. Summarized pipeline iteration. ϕ and θ represent the roll and pitch angles provided for computing the elevation displacement in the global frame for ELOFF.

intended. Also, even though it is theoretically possible to use low-resolution maps like Karlsruhe's EquatorStudio DSM, the elevation in a 30-meter region is provided with a high uncertainty due to many factors - including oscillations in the ground slope, buildings and vegetation - which implies in update steps that would not discard hypotheses.

For filtering the dynamic objects[13], we applied the Yolo detector (v5m)[14] in the RGB images of the left camera and filtered the projected LiDAR scan points that were inside the detections' bounding boxes. More than filtering, projecting the points to the RGB image frame allowed for colorizing the point cloud.

Using the reconstructed environment's point cloud, the DSM was built by projecting each point to a raster array using the transformation of Equation (4) presented in Section 3.2. In this array, each cell contains the maximum Z of the projected points. Some of the maps that resulted from this process are displayed in Figure 4.

4.2 Results

For comparing pose estimation methods, the metrics used were the Absolute Pose Error (APE) and the Relative Pose Error (RPE), which are generally split into translational and rotational error values. Those error metrics were computed using the EVO software [Grupp, 2017], which is a well-known tool for computing error metrics between trajectories and a reference. We computed metrics for the visual and laser-based odometry approaches used as input to the PF and for the 2D global localization.

The APE is computed using the relative transformation matrix between the estimated state S and the expected (annotated) state G, which is obtained using $T_S^G = (T_G^F)^{-1}(T_S^F)$, where $T_G^F \in SE(3)$ corresponds to the transformations between G and an arbitrary frame F, and $T_S^F \in SE(3)$ between S and F. $T_S^G \in SE(3)$, contains the linear and angular displacements from S to G, which are used for computing the translation and angle errors.

Following similar logic, the RPE compares the relative transformation of the frame S in a previous moment in time (S_{prev}) with the relative transformation from G at the same

	SLOPY(Ours)		DF-VO (Stereo)		Viso2	
Sequence	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
00	2.3	2.1	2.0	1.0	16.1	4.8
01	2.2	1.4	105.7	32.6	173.1	22.4
02	2.1	2.4	2.1	0.8	33.5	2.1
04	2.6	2.0	1.0	0.3	50.9	1.5
05	2.1	2.4	1.5	0.5	18.1	5.4
06	1.9	1.7	1.2	0.5	29.2	2.7
07	2.9	2.2	1.1	0.4	15.0	5.9
08	2.6	2.7	1.9	0.5	18.3	3.2
09	1.9	2.8	3.1	0.4	28.1	2.4
10	2.3	3.2	3.5	0.5	32.9	3.4
Average	2.3	2.3	12.3	3.8	41.52	5.38

Table 1. Comparison of the estimated odometry trajectories' relative position (t_{rel}, in %) and orientation (r_{rel}, in $deg/100m$) error metrics.

moment G_{prev}, so that the RPE pose matrix is computed as $T_S^G = (T_G^{G_{prev}})^{-1}(T_S^{S_{prev}})$. More information can be found in the EVO source code[15] and documentation[16].

The goal of the odometry approaches is to provide local estimates that are not globally corrected over time. Therefore, the appropriate metric used for comparison is the RPE, which basically computes the difference in the offset between the reference and the odometry. In the KITTI dataset, they recommend using window sizes above $100m$ for the comparison. In Table 1, we compared the translational and rotational RPE for SLOPY, DF-VO, and Viso2. The DF-VO and Viso2 results were provided by Zhan *et al.* [2021][17].

Regarding the global pose estimation - which is our main goal - the absolute position and orientation errors are gathered in Table 2. For computing those metrics, since the reference frame is the camera, the Z-axis points forward and the X-axis points to the left, which means that the Y component of the estimations and the reference were ignored and the rotation part was recomputed by setting the roll and yaw components to zero both to the reference and the estimations.

As for other localization methods, in the paper presented in Yan *et al.* [2019], the authors tested their localization algorithm using the KITTI 00, 05, 06, 07, 09, and 10 sequences

[13]The objects considered dynamic in our filtering are "car", "person", "bicycle", "motorcycle", "handbag", "train", "truck", and "bus".

[14]From this implementation https://github.com/ultralytics/yolov5

[15]Available at: https://github.com/MichaelGrupp/evo/blob/master/evo/core/metrics.py.

[16]Available at https://github.com/MichaelGrupp/evo/blob/master/notebooks/metrics.py_API_Documentation.ipynb

[17]At https://github.com/Huangying-Zhan/DF-VO

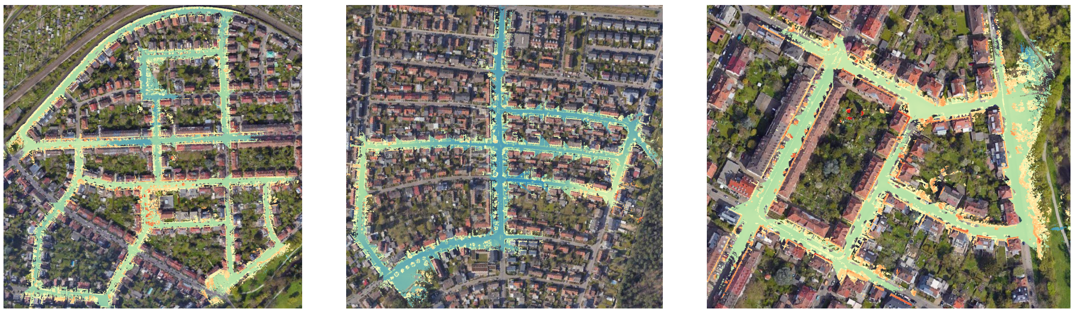

(a) 00 **(b)** 05 **(c)** 07

Figure 4. The digital surface maps of 3 of the sequences constructed using the reference trajectory.

Odometry	Uncorrected	ELOFF	GLOSM	Both
SLOPY	$40.6m$ - 6.0^o	$6.4m$ - 2.0^o	$3.6m$ - 0.9^o	$4.3m$ - 1.7^o
DF-VO	$81.7m$ - 5.3^o	$70.1m$ - 3.3^o	$69.5m$ - 2.46^o	$75.1m$ - 3.2^o
SLOPY*	$28.9m$ - 5.7^o	$5.8m$ - 2.0^o	$3.3m$ - 0.87^o	$4.5m$ - 1.7^o
DF-VO*	$11.3m$ - 5.6^o	$5.6m$ - 2.1^o	$4.1m$ - 1.2^o	$10.6m$ - 2.6^o

Table 2. The estimated trajectories' average 2D position (t_{abs}) and orientation (r_{abs}) error metrics computed on the KITTI sequences from 0 to 10. The methods indicated with * had their averages computed without sequence 01, in which DF-VO was unable to converge.

Sequence	Miller *et al.* [2021]	Brubaker *et al.* [2016] (Stereo)	**Ours**
00	$2.0m$	$2.2m$	$2.8m$
01	$2.0m$	$2.6m$	$6.5m$
02	$9.1m$	$4.3m$	$3.1m$
05	-	$3.7m$	$2.6m$
07	-	$1.7m$	$1.5m$
08	-	$6.0m$	$3.6m$
09	$7.2m$	$4.4m$	$3.6m$
10	-	$4.9m$	$3.2m$
Average	$5.1m$	$4.0m$	$3.4m$

Table 3. Comparison between the average 2D position error metrics of the proposed approach with related works. For the column of our approach, we selected the combination SLOPY with GLOSM, which yielded the best results.

but did not specify the exact error metrics, however, their claim is that - after convergence - their localization reach an error close to 10m. In Miller *et al.* [2021], they also tested their localization algorithm using the 00, 02 and 09 sequences. In Brubaker *et al.* [2016], the authors tested their method in all the initial sequences but failed to converge in sequences 04 and 06.

The mean trajectories' error metrics of those papers are displayed in Table 3.

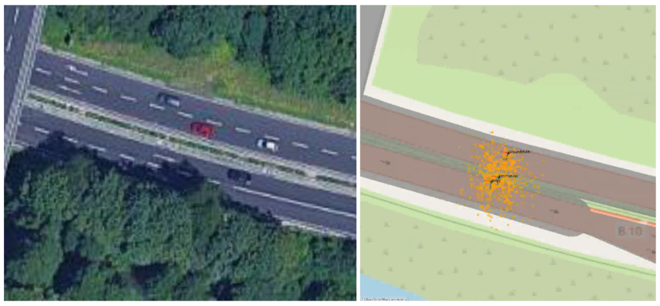
Regarding the processing time, the average processing time of the MCL predictions, GLOSM, and ELOFF update times are displayed in Table 4. For the sequences above, the experiments were conducted using 500 particles, but we benchmarked with more particles in order to verify the limits for real-time applications.

In addition, the videos of the experiments, as well as the estimated trajectories and tables of the experiments' metrics, are available in our paper's GitHub repository[18].

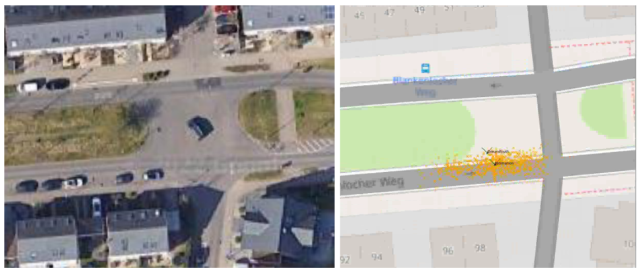
4.3 Analysis

Even though SLOPY implements basically point-to-plane ICP registrations with velocity estimations, it proved to be consistent even on a sequence (sequence 02) in which the vehicle is at a higher speed, which is in general an issue for odometry estimation, as happened with DF-VO and Viso2. We emphasize that the global localization method we propose in our paper can use odometry poses estimated using any sensor modality.

In practice, GLOSM's role in the pipeline was to ensure that the trajectory was contained reasonably well inside the navigable area. Its strength was in the parts of the trajectory with very narrow lanes or in trajectories with many curves. However, in the highway (sequence 01) the roads are larger, which implies a larger margin for the particles to spread and, therefore, less accurate estimations. Added to this fact, in some parts of the trajectory the driveable area was not correctly mapped. An instance of each of the mentioned issues is illustrated in Figure 5.



(a) Large roads.



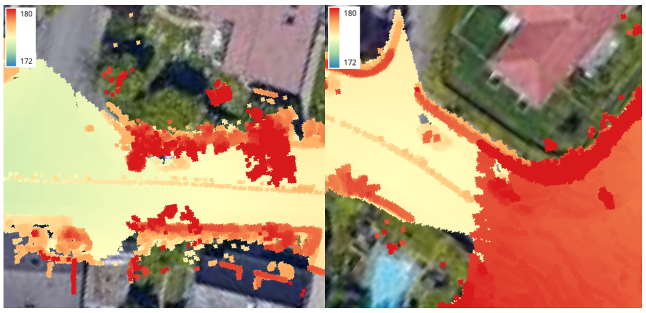
(b) Roads not fully covered.

Figure 5. The artifacts found in the GLOSM map.

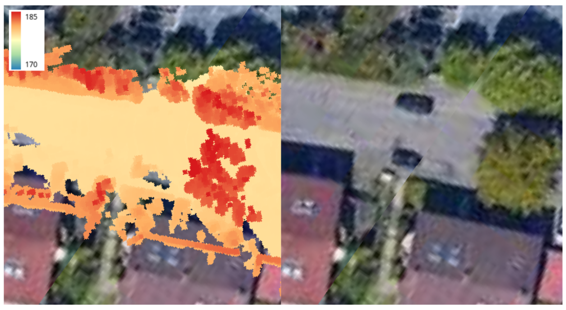
[18]Available at github.com/cabraile/GLOSM-ElOff

Num. Particles	**Prediction**	**GLOSM**	**ELOFF**	**Total**
500	$9ms \pm 3ms$	$42ms \pm 149ms$	$0.2ms \pm 0.7ms$	$51.2ms$
1000	$18ms \pm 5ms$	$79ms \pm 212ms$	$0.3ms \pm 0.2ms$	$97.3ms$
2000	$39ms \pm 12ms$	$157ms \pm 358ms$	$0.7ms \pm 2.3ms$	$193.7ms$

Table 4. The average and standard deviation of the processing time of the proposed method's steps.



(a) Dynamic artifacts **(b)** Elevation Oscillation



(c) Road Occlusion

Figure 6. The artifacts found in the reconstructed DSM. In (a) the top of a moving vehicle was not filtered during several frames, leading to the orange line in the DSM. (b) An abrupt change of elevation (more than 5m) causes the map discontinuity from the yellow to the red parts. (c) The tree canopy that was covering the street was mapped and is contained in the DSM.

The dynamic objects' filtering mitigated the artifacts from the constructed maps but did not remove completely their influence. Also, the height of the reference oscillated by a large margin in a slow period of time in some cases. Using the surface models built from a ground vehicle's LiDAR is not ideal: while filtering can help, aerial LiDAR imagery tends to be more accurate (can achieve a resolution lower than 10cm) and less prone to the objects' motion. ELOFF is also prone to issues when tall objects cover the road in the DSM, for instance, traffic signals or light poles that are curved towards the street. An example for each of the issues described above is displayed in Figure 6. Still, even with the presented limitations, the proposed method was able to converge.

Incorporating GLOSM to ELOFF in this set of experiments improved its estimations, but quantitatively the GLOSM with ELOFF approach was still not better than using GLOSM alone. In general, most of the sequences had a high presence of height artifacts in the same lane of the vehicle, which led the particles to converge on a parallel lane. In the sequences in which the artifacts were not prominent, the ELOFF helped in correcting the drift mainly for the cases where the vehicle takes too long to perform a turn. One of those cases is illustrated in Figure 7.

In DF-VO the average error of the fusion of GLOSM with ELOFF increased when compared with the estimates being

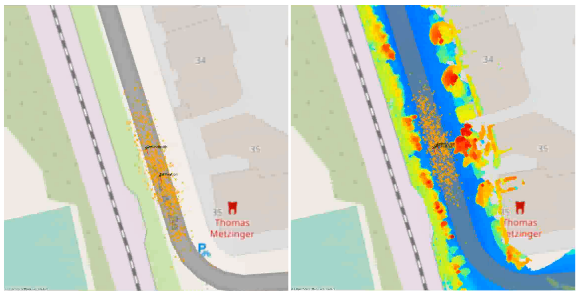


Figure 7. A situation where the vehicle took a long period of time before turning (moving from the top to the bottom). On the left, the GLOSM-estimated state and particles. On the right, the estimated state uses both GLOSM and ELOFF. This example was performed on Sequence 00 using SLOPY. Particles are represented by yellow crosses.

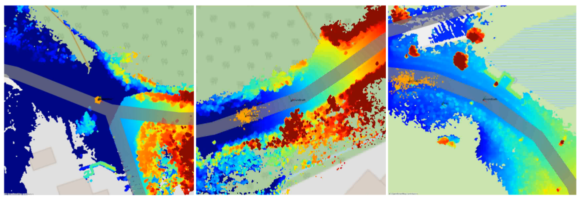


Figure 8. The sequence of frames (from the left to the right) that illustrate the failure in the convergence of DF-VO corrected with GLOSM and ELOFF. The altitude of the map is colorized so that the lowest point is blue and the highest point is red. Notice that right at the first frame (left figure), the height ramp changes abruptly and that DF-VO odometry was very delayed with respect to the reference.

performed individually. This happened because, in sequence 09, we had a combination of the DSM height oscillation artifacts allied with bad odometry estimates at the beginning. Those made the filter diverge very close to the start of the trajectory. This is illustrated in Figure 8.

When it comes to global localization, our approaches were capable of reducing the trajectory drift error by approximately $20m$ and localizing the agent with an average error below $5m$ without the usage of GNSS devices or high-precision maps and even performed better than the approaches used in our comparison. Besides the accuracy, the proposed approach is relatively fast: on average, using GLOSM and ELOFF together for updating 500 particles takes close to $50ms$ on an i7 8th generation processor, which means that the method can operate close to 20Hz, exceeds the regular input frequency of LiDAR scans, and is very close to the operation frequency of camera sensors usually used in the context of self-driving agents. Even more, the implementation provided is a Python-prototyped code and was not fully optimized so that the filter's speed can be enhanced further.

5 Conclusion

In this paper, we used both OSM and DSM information for performing 2D global pose estimation. The proposed method is relatively lightweight, was able to achieve a relatively low pose error if considering its modality, does not depend on sensor-specific data, and is open-source.

Considering the experiments conducted in the KITTI dataset, the driveable area constraint was the most relevant feature for reducing the trajectory drift. Yet, it is more effective on trajectories with more turns, while the DSM constraint helped to correct the drift for longer trajectories without turns.

Still, we believe that using the DSM as a constraint can be more effective if reconstructed from aerial unmanned vehicles since most of the effects of the elevation oscillation and dynamic objects would be solved. However, handling map occlusion by tree canopies or other higher objects that overlap the road is still an open issue that would happen even for more precisely collected DSM.

Also, even though we used GNSS only for initializing the filter, this dependency could be replaced by incorporating the localization pipeline methods that focus on providing an initial global pose when no prior global position knowledge is available, which is the goal of the approach proposed in Yan *et al*. [2019].

In future work, we intend to explore the localization using DEM instead of DSM, since trees and other tall objects' influence are not present in DEM. Also, we believe that estimating pitch and yaw from the DEM or DSM would require small changes in the framework, which is also to be explored in later work.

Declarations

Funding

This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil* (CAPES) – Finance Code 001.

Authors' Contributions

AP contributed to the conception of this study. Professors FO and VG supervised the project. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests

References

Amini, A., Rosman, G., Karaman, S., and Rus, D. (2019). Variational end-to-end navigation and localization. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May:8958–8964. DOI: 10.1109/ICRA.2019.8793579.

Bhattacharyya, P., Gu, Y., Bao, J., Liu, X., and Kamijo, S. (2017). 3D Scene Understanding at Urban Intersection Using Stereo Vision and Digital Map. In *IEEE Vehicular Technology Conference*, volume 2017-June. Institute of Electrical and Electronics Engineers Inc.. DOI: 10.1109/VTCSpring.2017.8108283.

Brubaker, M. A., Geiger, A., and Urtasun, R. (2016). Map-based probabilistic visual self-localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):652–665. DOI: 10.1109/TPAMI.2015.2453975.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237. DOI: 10.1177/027836491349129.

Grupp, M. (2017). evo: Python package for the evaluation of odometry and SLAM. Available at:{https://github.com/MichaelGrupp/evo}.

Hyuga, S., Ito, M., Iwai, M., and Sezaki, K. (2016). An online localization method for a subway train utilizing the barometer on a smartphone. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*. DOI: 10.1145/2996913.2996999.

Imperoli, M., Potena, C., Nardi, D., Grisetti, G., and Pretto, A. (2018). An Effective Multi-Cue Positioning System for Agricultural Robotics. *IEEE Robotics and Automation Letters*, 3(4). DOI: 10.1109/LRA.2018.2855052.

Larnaout, D., Gay-Bellile, V., Bourgeois, S., and Dhome, M. (2013). Vehicle 6-DoF localization based on SLAM constrained by GPS and digital elevation model information. In *2013 IEEE International Conference on Image Processing, ICIP 2013 - Proceedings*. DOI: 10.1109/ICIP.2013.6738516.

Mandel, C. and Laue, T. (2010). Particle filter-based position estimation in road networks using digital elevation models. In *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*. DOI: 10.1109/IROS.2010.5650755.

Miller, I. D., Cowley, A., Konkimalla, R., Shivakumar, S. S., Nguyen, T., Smith, T., Taylor, C. J., and Kumar, V. (2021). Any Way You Look at It: Semantic Crossview Localization and Mapping with LiDAR. *IEEE Robotics and Automation Letters*, 6(2). DOI: 10.1109/LRA.2021.3061332.

Przewodowski, A. and Osorio, F. S. (2022). A Monte Carlo particle filter formulation for mapless-based localization. *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 1782–1788. DOI: 10.1109/IV51971.2022.9827064.

Quack, T., Heßeler, F. J., and Abel, D. (2019). Fast real-time localization with sparse digital maps for connected automated vehicles in urban areas. In *IFAC-PapersOnLine*, volume 52, pages 366–371. Elsevier B.V.. DOI: 10.1016/j.ifacol.2019.09.059.

Radwan, N., Tipaldi, G. D., Spinello, L., and Burgard, W. (2016). Do you see the bakery? Leveraging geo-referenced texts for global localization in public maps. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:4837–4842. DOI: 10.1109/ICRA.2016.7487688.

Sarlin, P.-E., DeTone, D., Yang, T.-Y., Avetisyan, A., Straub, J., Malisiewicz, T., Bulo, S. R., Newcombe, R., Kontschieder, P., and Balntas, V. (2023). Orienternet: Visual localization in 2d public maps with neural matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21632–21642. Available at:https://openaccess.thecvf.com/content/CVPR2023/html/Sarlin_OrienterNet_

Visual_Localization_in_2D_Public_Maps_With_Neural_Matching_CVPR_2023_paper.html.

Yan, F., Vysotska, O., and Stachniss, C. (2019). Global Localization on OpenStreetMap Using 4-bit Semantic Descriptors. In *2019 European Conference on Mobile Robots, ECMR 2019 - Proceedings*. Institute of Electrical and Electronics Engineers Inc.. DOI: 10.1109/ECMR.2019.8870918.

Zhan, H., Weerasekera, C. S., Bian, J., Garg, R., and Reid, I. D. (2021). DF-VO: What Should Be Learnt for Visual Odometry? *ArXiv*, abs/2103.00933. DOI: 10.48550/arXiv.2103.00933.

Zhan, H., Weerasekera, C. S., Bian, J. W., and Reid, I. (2020). Visual Odometry Revisited: What Should Be Learnt? In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4203–4210. DOI: 10.1109/ICRA40945.2020.9197374.