Magnetic Field Mapping of Railway Lines with Graph SLAM

Benjamin Siebler*, Andreas Lehner*, Stephan Sand*, and Uwe D. Hanebeck†
*German Aerospace Center (DLR), Institute of Communications and Navigation
†Karlsruhe Institute of Technology (KIT), Intelligent Sensor-Actuator-Systems (ISAS) Laboratory
benjamin.siebler@dlr.de

***Abstract*—The earth's magnetic field along railway tracks is strongly distorted by magnetic material in the vicinity, e.g., steel in rails and reinforced concrete. The resulting magnetic distortions are persistent in time and characteristic for a certain part of the track. Thus, these distortions can be seen as fingerprints that enable localization when a map of the magnetic field is available. This is particularity interesting for areas where global navigation satellite system (GNSS) signals are not available, such as tunnels. Unfortunately, creating the magnetic map in a GNSS-denied area requires a position reference system that is most likely not available. This paper addresses this problem with a graph-based simultaneous localization and mapping (SLAM) algorithm that uses only odometer and magnetometer measurements. The key idea of the proposed algorithm is to use the magnetic field to detect loop closures and to calculate the relative transformation between different nodes in the pose-graph.**

The algorithm is evaluated based on a data set recorded with the advanced TrainLab of the Deutsche Bahn traveling on a track in Berlin. Results show that the graph SLAM algorithm together with the magnetic loop closure detection reduces and bounds the position error of the odometry.

I. Introduction

Magnetic field-based train localization has the potential to enable localization also in areas where global navigation satellite systems (GNSS) are not available without the need to install track-side devices dedicated for localization such as balises or radio beacons [1]. Magnetic localization is a fingerprinting method and requires a map that relates a certain magnetic fingerprint or pattern to the position where the pattern is located. Unfortunately, creating the map requires a position reference leading to a chicken and egg problem: Magnetic localization requires a map and map generation requires a reference position. This kind of problem is encountered in many applications, e.g., in robotics where a mobile robot has to create a map of an unknown environment for self-localization. Over the last years this led to the development of a large variety of simultaneous localization and mapping (SLAM) algorithms [2], [3], [4], [5].

Initially SLAM was based on Bayesian filters, such as extended Kalman filters [2] and particles filters [3]. Filter-based SLAM also has been applied to magnetic field mapping [6], [7]. These algorithms express the posterior density of the localization part of the SLAM problem with a particle filter and combine them with Gaussian processes (GPs) to represent the magnetic maps. For railway tracks with lengths of multiple kilometers this combination causes two issues. The first issue is related to the particle filter that represents the localization related posterior by a weighted set of particles. If a train travels back and forth on a long track, the time until it reaches the same position again, and therefore the time until a loop closure can be detected, is long. This results in a potentially high location uncertainty and a high number of particles to represent the uncertainty. The second issue is related to the GP for the magnetic field maps. A naïve implementation of the GP scales cubically with the amount of incorporated measurements. Even though there exist more efficient approximations for GPs, e.g., reduced-rank approximations [6], the complexity is still high when an accurate approximation of the GP is desired.

In this paper, we thus propose to not use filter-based methods but graph-based methods that are better suitable for the mapping of large-scale environments [4]. Graph SLAM poses the SLAM problem as an optimization problem, where the goal of the optimization is to find a sequence of locations that explain the available measurements as well as possible. To achieve this goal, a cost function is constructed and minimized. The cost function consists of multiple constraints, where each constraint is derived from the available measurements. Typically, constraints describe spatial relationships between different nodes in the graph. A common example for a constraint is obtained from an odometer measurement and expresses the distance between two consecutive nodes. Another constraint is introduced by loop closures. When a loop closure is detected, a spatial relation between two, not necessarily consecutive, nodes is added to the graph. How the loop closures are detected strongly depends on the used sensor setup. Popular methods for SLAM use lidar scans or camera images to identify already visited locations [8], [9]. In contrast to this, we propose the use of the magnetic field to detect the loop closures. The algorithm developed in this paper is inspired by the work in [10], where the magnetic field is used to detect loop closures for localizing a wheeled mobile robot. Our algorithm differs from the one in [10] in the way the loop closures are detected and how the magnetic field is represented. Instead of having a single node for each magnetometer measurements, we create a local map for each node from multiple odometer and magnetometer measurements. A local map is a dense representation of the magnetic field around the corresponding node, e.g., the magnetic field of the last 100 m of track before the node was created. In this way we use a dense representation of the magnetic field for loop closure detection while keeping

the number of nodes, that are optimized, low. Furthermore, the optimization algorithm is specifically adapted to and evaluated in the railway environment.

II. Graph SLAM

Graph-based SLAM algorithms formulate the SLAM problem as an optimization problem. Therefore, a cost function has to be defined. Here the cost function is the full posterior probability density function (pdf) of a sequence of train poses $x_{0:k}$

$$p(x_{0:k}|\mathcal{Z},\mathcal{Y}) \ , \tag{1}$$

where $x_{0:k}$ is a vector containing the poses and $\mathcal{Z}$ and $\mathcal{Y}$ are two sets of observations. A pose x_i in the vector $x_{0:k}$ is here simply the scalar along-track position that describes where the train is on the track. Set $\mathcal{Z}$ contains observations $z_{ij} \in \mathcal{Z}$ that describe how two along-track positions x_i and x_j with $i \neq j$ in the vector $x_{0:k}$ are related to each other. This kind of measurements can be obtained from an odometer that measures the distance the train has driven between two positions or, as will be explained in more detail in the next sections, this information can be also obtained from the magnetic field. In contrast to the set $\mathcal{Z}$, the set $\mathcal{Y}$ contains absolute information about a certain position. Therefore, an element y_i of $\mathcal{Y}$ is only related to a single position x_i. In the following, we will refer to a position x_i also as node and observations will be referred to as edges. The meaning of this will become clearer after the derivation of the cost function. The goal of the optimization is to find the maximum a-posteriori (MAP) estimate

$$\hat{x}_{0:k} = \underset{x_{0:k}}{\arg\max}\, p(x_{0:k}|\mathcal{Z},\mathcal{Y}) \ . \tag{2}$$

To perform the optimization, it is beneficial to decompose the posterior into a more convenient form. First, Bayes' law is applied to the posterior to obtain

$$p(x_{0:k}|\mathcal{Z},\mathcal{Y}) \propto p(\mathcal{Z},\mathcal{Y}|x_{0:k})p(x_{0:k}) \ . \tag{3}$$

Note, the right hand side above is only proportional to the posterior but since we only seek the maximum we can ignore the normalization constant. Second, the right hand side in (3) is further decomposed into

$$\begin{aligned} p(\mathcal{Z},\mathcal{Y}|x_{0:k})p(x_{0:k}) &= p(\mathcal{Y}|x_{0:k})p(\mathcal{Z}|x_{0:k})p(x_{0:k}) \\ &= p(x_{0:k}) \prod_{i\in\mathcal{C}_\mathcal{Y}} p(y_i|x_i) \prod_{\{i,j\}\in\mathcal{C}_\mathcal{Z}} p(z_{i,j}|x_i,x_j) \ , \end{aligned} \tag{4}$$

assuming independence between all the observations. The elements of set $\mathcal{C}_\mathcal{Z}$ are pairs of node indices for which relative observations are contained in $\mathcal{Z}$ and set $\mathcal{C}_\mathcal{Y}$ contains the indices of positions for which absolute observations exist. To not overload the notation, only one relative observation per pair of nodes and one absolute observation per node is considered here but the extension to multiple observations is straightforward.

The last two terms in (4) are the product of the likelihoods of the individual observations. For each of the observations the likelihood is considered to be Gaussian. For the relative observations in $\mathcal{Z}$ the likelihood is

$$p(z_{i,j}|x_i,x_j) = \mathcal{N}\left(z_{i,j};\tilde{z}(x_i,x_j),\Omega_{ij}^{-1}\right) \ , \tag{5}$$

where Ω_{ij}^{-1} is the variance of the measurement noise and the mean is given by the predicted observation $\tilde{z}(x_i,x_j)$. The likelihood of the absolute observations in $\mathcal{Y}$, and therefore its mean $\tilde{y}(x_i)$, depends only on a single position

$$p(y_i|x_i) = \mathcal{N}\left(y_i;\tilde{y}(x_i),\Omega_i^{-1}\right) \ , \tag{6}$$

where Ω_i^{-1} describes once more the measurements noise variance. For the prior pdf $p(x_{0:k})$ we assume that we only have information about the first node so we simply set it to

$$p(x_0) = \mathcal{N}(x_0;\tilde{x}_0,\Omega_0^{-1}) \ . \tag{7}$$

Setting the precision Ω_0 to a high value anchors the node at mean value $\tilde{x}_0$ of the prior. This mean can be obtained, e.g., from GNSS before the train enters a tunnel, or it can be set simply to zero to define the origin of the track coordinate system when no absolute position information is available.

For optimization, we simplify the cost function $p(\mathcal{Z},\mathcal{Y}|x_{0:k})p(x_{0:k})$ defined in (4)-(7) by first taking the logarithm, removing all constant terms, and then multiplying the result by -2. The multiplication with -2 turns the maximization problem into a minimization problem

$$\hat{x}_{0:k} = \underset{x_{0:k}}{\arg\min}\, \mathrm{c}(\mathrm{x}_{0:\mathrm{k}}) \ , \tag{8}$$

with the cost function

$$\mathrm{c}(\mathrm{x}_{0:\mathrm{k}}) = \mathrm{e}_\mathrm{p}(x_0)^2\Omega_0 + \sum_{i\in\mathcal{C}_\mathcal{Y}} \mathrm{e}_{\mathrm{a},i}(x_{0:k})^2\Omega_i + \sum_{\{i,j\}\in\mathcal{C}_\mathcal{Z}} \mathrm{e}_{\mathrm{r},ij}(x_{0:k})^2\Omega_{ij} \ . \tag{9}$$

In (9) the functions $\mathrm{e}_\mathrm{p}(\cdot)$, $\mathrm{e}_{\mathrm{a},i}(\cdot)$, and $\mathrm{e}_{\mathrm{r},ij}(\cdot)$ are the error functions related to the prior, absolute and relative observations given by

$$\mathrm{e}_\mathrm{p}\left(x_{0:k}\right) = x_0 - \tilde{x}_0 \tag{10}$$

$$\mathrm{e}_{\mathrm{a},i}(x_{0:k}) = y_i - \tilde{y}(x_i) \tag{11}$$

$$\mathrm{e}_{\mathrm{r},ij}(x_{0:k}) = z_{i,j} - \tilde{z}(x_i,x_j) \ . \tag{12}$$

The cost function $\mathrm{c}(\mathrm{x}_{0:\mathrm{k}})$ can be visually represented by a graph, where each edge represents a constraint given by one term of the cost function and each node is a pose x_i. An example of a graph, as it is encountered in the railway environment, is shown in Fig. 1, where the different types of edges are highlighted with different colors. To better see the structure of the graph and to avoid overlap of nodes and edges, we introduce a second dimension for visualization that indicates how often the train changes its driving direction before the node is created. The graph in Fig. 1 shows the example of a train running forth and back on a single track. Two neighboring nodes are always connected with an odometer observation (blue edge). In addition, two non-consecutive nodes can be

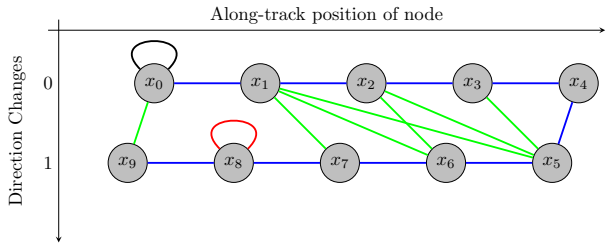

Figure 1. Example of a small graph with color coded edges. Black loops indicate that prior information is available for that node and red loops indicate an absolute observation. Blue and green edges correspond to a relative observation between two nodes. In this paper, relative observations are either related to an odometer measurement (blue edges) or a magnetic loop closure (green edges).

connected with a loop closure (green edges). Prior information is indicated by a black loop from a node to itself. Similarly, absolute information is represented by a red loop.

III. Graph Generation

In the previous section, the general idea of the graph-based SLAM approach was explained on a rather abstract level. For clarity, this section explains how the graph is generated and how the different observations used in this paper are obtained.

A. Node Creation

To keep the number of nodes at a reasonable level, we do not create a node with each new observation from the odometer or magnetometer but instead we create a node every time the train has traveled a certain distance relative to the last created node. At creation time, the node position is initialized with the position of the previous node plus the traveled distance measured with the odometer. Even though nodes are created only sparsely, all observations are considered in the SLAM algorithm and are used to create local magnetic field maps at the moment a new node is created. Each node therefore has its own local map. As will be explained in the following, these maps play a crucial role for loop closure detection.

B. Odometer Edges

As shown in the example graph in Fig. 1, there is an edge between consecutive nodes. The observations $z_{i,j}$ for edges between consecutive nodes are here obtained from an odometer. The odometer directly measures the relative position, thus the error function for odometer edges is linear

$$\mathrm{e}_{\mathrm{r},ij}(x_{0:k}) = z_{i,j} - (x_i - x_j) \ . \tag{13}$$

In railways, the odometer observations are typically obtained from integrating the speed of a wheel speed sensor or a Doppler radar. Under nominal conditions, these sensors work quite well but there are also known issues that degrade their performance. For axle mounted rotary encoders, slipping and sliding of the wheel can lead to erroneous speed measurements and Doppler radar-based speed sensors are susceptible to environmental conditions such as snowfall.

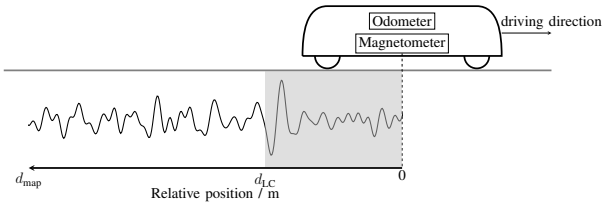

Figure 2. Example of a local map. The position is measured always relative to the current magnetometer position during node creation. Note, the map always describes the magnetic field on the part of the track that was passed before the node creation.

C. Loop Closure Edges

Loop closures are relative edges between non-consecutive nodes. A loop closure can be detected when a train revisits a position on the track. The benefit of a loop closure edge is that it links freshly added nodes, which have a high uncertainty, to old nodes, which are more certain. Loosely speaking, these links between new and old nodes constrain the overall trajectory of the train over the track and therefore can reduce the position error compared to the pure odometer.

The main contribution of this paper lies in the loop closure detection. Here, the loop closures are detected based on local magnetic field maps. Each node has its own map that describes the magnetic field in its surroundings. A magnetic field map is a function that maps the along-track position to the magnetic vector field at that position. For local maps the position is always relative w.r.t. the position of the corresponding node. Whenever a new node is added to the graph, the local maps are used to detect loop closures. The detection consists of three steps

- Find the set $\mathcal{L}$ of nodes that are in a predefined search radius around the freshly created node.
- Compare the magnetic field recorded on the last d_{LC} meters preceding the new node's position with the local maps of all nodes in $\mathcal{L}$ and find the position within these maps where it best fits.
- Add an edge between the current node i and a node $j \in \mathcal{L}$ when the similarity between the current magnetic field and the local map is above a defined threshold T_{LC}. The relative observation $z_{i,j}$ of the edge is given by the relative position where the current magnetic field fits best to the local map of node j. The error function for loop closure edges is identical to the one for odometer edges given by (13).

In the following, the map creation and the similarity calculation is explained in more detail:

1) Local Maps: For the local maps, the magnetometer and odometer measurements of the train are stored in a buffer. From that buffer, a local map is created whenever a new node is created. In the map creation, the buffered speed is integrated, starting with the newest available measurement. The integrated speed is then basically the along-track position measured relative to the freshly created node. By associating the relative positions to the magnetometer measurements in

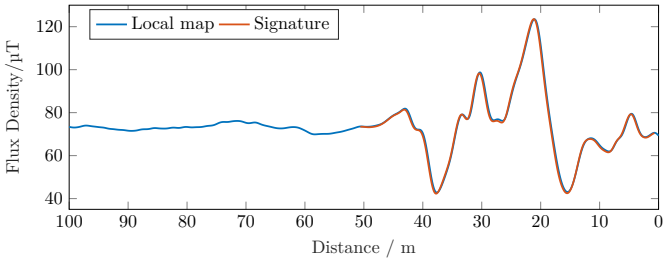

Figure 3. Match found from the correlation of a 50 m long magnetic signature (red line) with a 100 m long local map (blue line). This example shows only the matching for a single component of the vector field but for finding the shift all components were considered.

the buffer a first map is obtained. The distance between two relative positions from the speed integration varies depending on the train's motion during the measurements. Since this would complicate further processing, the unequally spaced map is interpolated on an equidistant grid. An example of a local magnetic map is shown in Fig. 2. The map length d_{map} can be chosen freely but with an increasing length also the speed integration error increases. Thus, the length of the map is a trade-off between maximizing the number of loop closures between nodes and the accuracy and reliability of the loop closures.

2) Similarity Calculation: If a new node is created, we also create its local map and store it with the node. Then we cut out a segment with length d_{LC} from the map that we call magnetic signature. The signature is cut from the start of the map as indicated by the gray area in Fig. 2. In general, the signature length is smaller than the length of the map $d_{\text{map}} > d_{\text{LC}}$. This ensures that loop closures can be found between nodes that are not at the exact same along-track position. As similarity measure between the signature and a local map of a node in $\mathcal{L}$, the correlation coefficient $\rho(d)$ is used

$$\rho(d) = \frac{\sum_{k=1}^{N} (m_{k,d} - \bar{m}_d)(s_k - \bar{s})}{\sqrt{\sum_{k=1}^{N} (m_{k,d} - \bar{m}_d)^2} \sqrt{\sum_{k=1}^{N} (s_k - \bar{s})^2}} , \tag{14}$$

where s_k is the k-th sample of the signature and $m_{k,d}$ is the k-th sample of a local map cutout between the relative position d to $d + d_{\text{LC}}$. In the correlation coefficient, $\bar{s}$ and $\bar{m}_d$ represent the sample mean of the signature and map cutout. Per definition the correlation coefficient is limited to values between ± 1. Because the map is longer than the signature, the correlation coefficient is evaluated for all possible positions d of the signature in the local map. In addition, the correlation coefficient is calculated separately for each of the three components of the magnetic vector field. To make the loop closure detection more robust in the presence of erroneous magnetometer measurements, the detection is based on the combination of the correlation coefficients of multiple components of the magnetic vector field. In total four combinations of correlation coefficients are considered. The first three are the sum of the coefficients obtained from two different magnetic field components and the fourth combination is the sum over all components. Each combination is then divided by the number of values in the sum such that that result is limited again to ± 1. We then find the highest value ρ_{max} of all combinations and store the corresponding position d_{max}, at which the map cutout starts. If $\rho_{\text{max}} > T_{\text{LC}}$ is fulfilled, a loop closure is detected between the newly created node and the node that corresponds to the local map used in the correlation. From d_{max}, the relative position $z_{i,j}$ between the two nodes is calculated and the loop closure edge is inserted into the graph.

An example for a local map and a signature for which a loop closure was detected is shown in Fig. 3. In Fig. 3, the signature was shifted by d_{max} found from the correlation with the local map. Note, when a train is running back and forth on a track, its driving direction during creation of the local maps and signatures has to be accounted for in the calculation of the correlation coefficient and in the relative position $z_{i,j}$ obtained from it.

D. Absolute Edges

Absolute edges will not be used in this paper but for completeness we wanted to mention how the corresponding observations can be obtained in the railway domain. Two possible sources for absolute position information are balises and GNSS. Balises are placed in the middle of the track between the two rails and are comparable to an RFID tag. When a train passes over a balise, the balise transmits information to the train related to its position. The observations retrieved from a balise are already in the 1D domain of the track coordinate system and can be used directly to create an absolute edge. In contrast, the observations obtained from a GNSS receiver describe the position of the train in the three-dimensional space, e.g., in the ECEF coordinate system. To create an edge for GNSS observations, the observations first have to be related to the 1D track coordinate system. This requires a map of the track network that maps each along-track position to its position in three-dimensional space. In this paper, we do not use absolute edges because we are mainly interested on how the proposed SLAM algorithm performs in areas without a position reference system such as tunnels.

IV. Optimization Algorithm

Now that the cost function and the graph generation was introduced, we can have a look on the optimization algorithm. Inspecting (9) shows that finding the MAP estimate of the trajectory is a weighted least-squares problem. For most SLAM problems the corresponding least-squares problem will be nonlinear and requires an iterative optimization, e.g., with the Gauss-Newton method [5]. In the special case considered in this paper, the least-squares problem is linear due to the linear error functions $\mathrm{e}_{\text{p}}(\cdot)$ and $\mathrm{e}_{\text{r},ij}(\cdot)$ defined in (10) and (13). The solution therefore can be calculated directly without iterations. To keep our optimization algorithm flexible and to allow for future nonlinear error functions, in the following we derive the equations for the Gauss-Newton algorithm for

general error functions. Furthermore, we keep the possibility to account for absolute information in the optimization.

For the derivation, the error functions are represented by their Taylor series. The Taylor series of the different functions are given by

$$\begin{aligned}
\mathrm{e}_{\mathrm{p}}^{\mathrm{l}}(x_{0:k}) &= \mathrm{e}_{\mathrm{p}}(x_{0:k}^*) + \mathbf{J}_{\mathrm{p}}(x_{0:k} - x_{0:k}^*) \\
\mathrm{e}_{\mathrm{a},i}^{\mathrm{l}}(x_{0:k}) &= \mathrm{e}_{\mathrm{a},i}(x_{0:k}^*) + \mathbf{J}_{\mathrm{i}}(x_{0:k} - x_{0:k}^*) \\
\mathrm{e}_{\mathrm{r},ij}^{\mathrm{l}}(x_{0:k}) &= \mathrm{e}_{\mathrm{r},ij}(x_{0:k}^*) + \mathbf{J}_{\mathrm{ij}}(x_{0:k} - x_{0:k}^*) \ , \qquad (15)
\end{aligned}$$

where $x_{0:k}^*$ is an arbitrary operating point and $\mathbf{J}_{\mathrm{p}}$, $\mathbf{J}_{\mathrm{i}}$, and $\mathbf{J}_{\mathrm{ij}}$ are the Jacobians of the error functions w.r.t. the vector $x_{0:k}$ evaulated at $x_{0:k}^*$

$$\begin{aligned}
\mathbf{J}_{\mathrm{p}} &= \nabla_{x_{0:k}}^{\top} \mathrm{e}_{\mathrm{p}}(x_{0:k})|_{x_{0:k}=x_{0:k}^*} \\
\mathbf{J}_{\mathrm{i}} &= \nabla_{x_{0:k}}^{\top} \mathrm{e}_{\mathrm{a},i}(x_{0:k})|_{x_{0:k}=x_{0:k}^*} \\
\mathbf{J}_{\mathrm{ij}} &= \nabla_{x_{0:k}}^{\top} \mathrm{e}_{\mathrm{r},ij}(x_{0:k})|_{x_{0:k}=x_{0:k}^*} \ . \qquad (16)
\end{aligned}$$

Since the error functions are scalar, the Jacobians are row vectors of dimension $k+1$ and the product with the vector $\Delta x_{0:k} = x_{0:k} - x_{0:k}^*$ is a scalar. Note, here the prior and relative error functions are linear and their Taylor series are exact and the Jacobians are constant. For nonlinear error functions this is not the case and the Jacobians have to be updated when the operating point is changed. Now the Taylor series representations of the error functions are plugged into the cost function in (9) which yields

$$\begin{aligned}
\mathrm{c}^{\mathrm{l}}(x_{0:k}) &= \mathrm{e}_{\mathrm{p}}^{\mathrm{l}}(x_{0:k})^2 \Omega_0 + \sum_{i \in \mathcal{C}_{\mathcal{Y}}} \mathrm{e}_{\mathrm{a},i}^{\mathrm{l}}(x_{0:k})^2 \Omega_i \\
&\quad + \sum_{\{i,j\} \in \mathcal{C}_{\mathcal{Z}}} \mathrm{e}_{\mathrm{r},ij}^{\mathrm{l}}(x_{0:k})^2 \Omega_{ij} \\
&= \mathrm{c}_{\mathrm{p}}(x_{0:k}) + \mathrm{c}_i(x_{0:k}) + \mathrm{c}_{ij}(x_{0:k}) \ . \qquad (17)
\end{aligned}$$

The least-squares solution is obtained by setting the gradient of (17) to zero

$$\nabla_{x_{0:k}} \mathrm{c}^{\mathrm{l}}(x_{0:k}) = \nabla_{x_{0:k}} \left[\mathrm{c}_{\mathrm{p}}(x_{0:k}) + \mathrm{c}_i(x_{0:k}) + \mathrm{c}_{ij}(x_{0:k}) \right] = \mathbf{0}. \qquad (18)$$

The gradients of the individual parts of (18) are given by

$$\begin{aligned}
\nabla_{x_{0:k}} \mathrm{c}_{\mathrm{p}}(x_{0:k}) &= 2\Omega_0 \mathbf{J}_{\mathrm{p}}^{\top} \mathrm{e}_{\mathrm{p}}(x_{0:k}^*) + 2\Omega_0 \mathbf{J}_{\mathrm{p}}^{\top} \mathbf{J}_{\mathrm{p}} \Delta x_{0:k} \\
\nabla_{x_{0:k}} \mathrm{c}_i(x_{0:k}) &= \sum_{i \in \mathcal{C}_{\mathcal{Y}}} (2\Omega_i \mathbf{J}_{\mathrm{i}}^{\top} \mathrm{e}_{\mathrm{a},i}(x_{0:k}^*) + 2\Omega_i \mathbf{J}_{\mathrm{i}}^{\top} \mathbf{J}_{\mathrm{i}} \Delta x_{0:k}) \\
\nabla_{x_{0:k}} \mathrm{c}_{ij}(x_{0:k}) &= \sum_{\{i,j\} \in \mathcal{C}_{\mathcal{Z}}} (2\Omega_{ij} \mathbf{J}_{\mathrm{ij}}^{\top} \mathrm{e}_{\mathrm{r},ij}(x_{0:k}^*) + 2\Omega_{ij} \mathbf{J}_{\mathrm{ij}}^{\top} \mathbf{J}_{\mathrm{ij}} \Delta x_{0:k}) \ . \qquad (19)
\end{aligned}$$

Equation (19) can be expressed as a system of linear equations

$$-\mathbf{b} = \mathbf{H} \Delta x_{0:k} \ , \qquad (20)$$

with the vector

$$\begin{aligned}
\mathbf{b} &= 2\Omega_0 \mathbf{J}_{\mathrm{p}}^{\top} \mathrm{e}_{\mathrm{p}}(x_{0:k}^*) + \sum_{i \in \mathcal{C}_{\mathcal{Y}}} 2\Omega_i \mathbf{J}_{\mathrm{i}}^{T} \mathrm{e}_{\mathrm{a},i}(x_{0:k}^*) \\
&\quad + \sum_{\{i,j\} \in \mathcal{C}_{\mathcal{Z}}} 2\Omega_{ij} \mathbf{J}_{\mathrm{ij}}^{T} \mathrm{e}_{\mathrm{r},ij}(x_{0:k}^*) \\
&= \mathbf{b}_{\mathrm{p}} + \mathbf{b}_{\mathrm{a}} + \mathbf{b}_{\mathrm{r}} \qquad (21)
\end{aligned}$$

and the matrix

$$\begin{aligned}
\mathbf{H} &= 2\Omega_0 \mathbf{J}_{\mathrm{p}}^{T} \mathbf{J}_{\mathrm{p}} + \sum_{i \in \mathcal{C}_{\mathcal{Y}}} 2\Omega_i \mathbf{J}_{\mathrm{i}}^{T} \mathbf{J}_{\mathrm{i}} + \sum_{\{i,j\} \in \mathcal{C}_{\mathcal{Z}}} 2\Omega_{ij} \mathbf{J}_{\mathrm{ij}}^{T} \mathbf{J}_{\mathrm{ij}} \\
&= \mathbf{H}_{\mathrm{p}} + \mathbf{H}_{\mathrm{a}} + \mathbf{H}_{\mathrm{r}} \ . \qquad (22)
\end{aligned}$$

To minimize the cost function, we now solve for $\Delta x_{0:k}$ and then calculate the estimated absolute value of $x_{0:k}$ from it

$$\hat{x}_{0:k} = x_{0:k}^* - \mathbf{H}^{-1} \mathbf{b}. \qquad (23)$$

In the linear case, we can set the operating point $x_{0:k}^*$ to an arbitrary value, e.g., the zero vector, and will obtain always the unique least-squares solution after the first iteration. For nonlinear problems, multiple iterations of (23) are required. After each iteration, the operating point is set to the newest estimate $x_{0:k}^* = \hat{x}_{0:k}$. As already mentioned, for nonlinear functions the Jacobians depend on the operating point and matrix $\mathbf{H}$ and vector $\mathbf{b}$ have to be recalculated in each iteration.

A. Complexity Considerations

The matrix $\mathbf{H}$, that is inverted during optimization, has dimension $(k+1) \times (k+1)$. For large vectors $x_{0:k}$ this can result in a high computational complexity, especially when nonlinear observations are considered and the inverse has to be recalculated for multiple iterations. In the most naïve implementation this would have a complexity of $\mathcal{O}((k+1)^3)$. Fortunately, it can be shown that matrix $\mathbf{H}$ is sparse by construction, which allows to use the method also for a large number of nodes. To see this, we have a look at the structure of the individual terms in (22).

The prior edge contributes to $\mathbf{H}$ over the matrix

$$\mathbf{H}_{\mathrm{p}} = 2\Omega_0 \mathbf{J}_{\mathrm{p}}^{T} \mathbf{J}_{\mathrm{p}} \ , \qquad (24)$$

with Jacobian

$$\mathbf{J}_{\mathrm{p}} = \begin{bmatrix} \frac{\partial \mathrm{e}_{\mathrm{p}}(x_{0:k})}{\partial x_0}|_{x_{0:k}=x_{0:k}^*} & 0 & \cdots & 0 \end{bmatrix} \ , \qquad (25)$$

which has exactly one non-zero component. The product of $\mathbf{J}_{\mathrm{p}}$ with its transpose, and as a result also $\mathbf{H}_{\mathrm{p}}$, only has one non-zero component at the first position of its diagonal. Due to (25), the corresponding vector $\mathbf{b}_{\mathrm{p}}$ is also sparse. For the absolute observations the Jacobian matrix has a similar structure, which means that $\mathbf{H}_{\mathrm{a}}$ and $\mathbf{b}_{\mathrm{a}}$ are sparse. For $\mathbf{H}_{\mathrm{a}}$ the non-zero entries are the diagonal elements that correspond to a node with an absolute observation.

The relative observations introduce links between different nodes. In $\mathbf{H}$ this is represented by non-diagonal elements. This can be seen once more from the structure of the Jacobian. In $\mathbf{J}_{\mathrm{ij}}$ only the elements associated to the derivative w.r.t. x_i and x_j are non-zero. The outer product $\mathbf{J}_{\mathrm{ij}}^{\top} \mathbf{J}_{\mathrm{ij}}$ of the Jacobian for one relative observation thus has four non-zero elements. Two at the diagonal and two at the off-diagonal that correspond to x_i and x_j. This shows that both $\mathbf{b}$ and $\mathbf{H}$ are sparse and computational efficient methods for solving the linear system in (20) are applicable.

Figure 4. (top) Advanced TrainLab of the Deutsche Bahn. The advanced TrainLab is a diesel-electric train with a distributed traction concept. (bottom) Magnetometer array with five sensors. The different sensors are highlighted with circles. In this paper the sensor in the top middle (green circle) is used.

Figure 5. Test track between Berlin Grunewald and Halensee (red line) on which the measurements were recorded. The start and end position of the train in the evaluation is marked by the green dot. Image data: Google Earth

V. Evaluation

In this section, the proposed SLAM algorithm is evaluated with measurements recorded by sensors mounted on the advanced TrainLab of the Deutsche Bahn while it was driving on a track network in Berlin. In the following, first the measurement setup shown in Fig. 4 and the data collection process is described, and second the results of the SLAM algorithm are discussed.

A. Measurement Setup and Data Collection

The hardware setup during the measurements consists of a low-cost Kionix KMX62 magnetometer, a Septentrio GNSS receiver, and a rotary encoder from DEUTA. Additional information about the measurements and the setup can be found in [11]. In the SLAM algorithm, only the magnetometer and encoder data are considered. The observations from the GNSS receiver serve only as ground truth to assess the error of the SLAM algorithm. The magnetometer measurements were recorded at a rate of 200 Hz to ensure a dense sampling of the magnetic field also when the train is driving at higher speeds.

The used magnetometer was part of a magnetometer array mounted underneath the train. An image of the advanced TrainLab and the magnetometer is shown in Fig. 4. The magnetometer used for SLAM is marked with a green circle and was mounted in the middle of the train at a height of $\approx 42\,\mathrm{cm}$ above the rails.

In the evaluation, we only consider a short part of our measurements in which the train ran on a single track, as the SLAM algorithm does not yet take multiple tracks into account. The test track on which the measurements for the evaluation were recorded is shown in Fig. 5 and has a length of $1.68\,\mathrm{km}$. The data used in the evaluation is divided into four runs. In each run the train drives from the middle of the test track, marked by the green dot in Fig. 5, to Grunewald, waits there a few minutes, and then drives all the way to Halensee and then back to the starting point. Due to some operational requirements, the train did not stop at Halensee but kept running towards Tempelhof (to the right in Fig. 5) and it took a couple of minutes until the train arrived back at the test track. To keep the data limited to the test track and a single track scenario, the part of the data between Halensee and Tempelhof was removed based on the recorded GNSS positions, leading to a discontinuity in the data. The discontinuity is handled by constructing a virtual odometer measurement from GNSS measurements at the discontinuity. When the end of the test track at Halensee is reached, the virtual odometer measurement is inserted into the measurements from the rotary encoder. Besides the discontinuities at Halensee, the data of the four runs is continuously recorded.

B. Parameters and Initialization

Before the SLAM algorithm is applied to the data of the four runs described in the previous section, some parameters have to be set. For loop closure detection the length of the local maps is set to 100 m and the signature length is 50 m. The distance between two nodes to be considered in the loop closure detection is set to 100 m. The threshold the correlation coefficient has to exceed before a loop closure is added to the graph is set to 0.97.

A new node is created as soon as the distance to the last node, obtained from the odometer, exceeds $d_{\text{node}} = 25\,\mathrm{m}$. Note, this does not mean that nodes are spaced equidistantly due to the low odometry sampling rate of 1 Hz. For example, during the measurements the train was traveling with a speed up to $15\,\mathrm{m\,s^{-1}}$. When now the distance to the last created node is just below d_{node}, it can happen that a new node is created at a distance of almost 40 m. If needed, one could use also higher sampling rates for the odometry to get a more even spacing of the nodes adding almost no computational complexity to the SLAM algorithm.

In the cost function and hence in the optimization, the different types of edges are weighted differently. The odometry edges between two consecutive nodes are weighted by $\Omega_{\text{odo}} = (1\,\mathrm{m})^{-2}$ and the loop closure edges are weighted

Table I
POSITION ERROR STATISTICS FOR A SINGLE RUN

	Odometry		SLAM	
Run Number	RMSE/m	Max Error/m	RMSE/m	Max Error/m
1	1.85	4.52	1.00	3.13
2	1.95	3.77	1.07	2.43
3	2.20	4.66	1.24	2.65
4	1.89	3.90	1.32	2.95

Table II
POSITION ERROR STATISTICS FOR CONSECUTIVE RUNS

	Odometry		SLAM	
Run sequence	RMSE/m	Max Error/m	RMSE/m	Max Error/m
1	1.85	4.52	1.00	3.13
1,2	3.11	6.20	1.04	3.11
1,2,3	4.67	10.20	1.10	3.06
1,2,3,4	6.52	12.87	1.06	3.12

with $\Omega_{\text{lc}} = (0.1\,\text{m})^{-2}$. The first node is anchored at the true along-track position on the test track by adding a prior edge with high precision Ω_{prior}. The value of Ω_{prior} is somewhat arbitrary and has to be large enough to avoid that the starting node is shifted during optimization. The prior edge ensures that the coordinate system of the SLAM algorithm is aligned with the track coordinate system and allows us to compare the result of SLAM to the ground truth obtained from GNSS. It is important to emphasize that knowing the starting position is by no means necessary for the SLAM algorithm and is only required when the SLAM result has to be related to another coordinate system.

C. Results

In Table I and Table II, the position root mean square error (RMSE) and the maximum absolute error are listed for different scenarios. In Table I, the errors are shown when the SLAM algorithm is evaluated separately for each of the four runs. In contrast, Table II shows the results when the algorithm is evaluated for multiple consecutive runs. The values in both tables are calculated using the position errors of the nodes. To obtain the position error for each node, the GNSS position, recorded at the time of node creation, is matched to a map of the test track which yields the ground truth along-track position. From the ground truth along-track position, the position error is calculated by comparing it to the positions of the nodes obtained from the unaided odometry and the positions optimized with our SLAM algorithm.

The results in Table I show that for all four runs the SLAM algorithm can lower the RMSE and the maximum error considerably. To understand where this improvement is coming from, we will have a look on the pose-graph after the first run shown in Fig. 6. In Fig. 6, the x-axis shows the along-track position of the nodes and the y-axis shows how often the train has changed its driving direction at the time of node creation. The first node is indicated by a prior edge in form of a black loop. From the starting position, the train

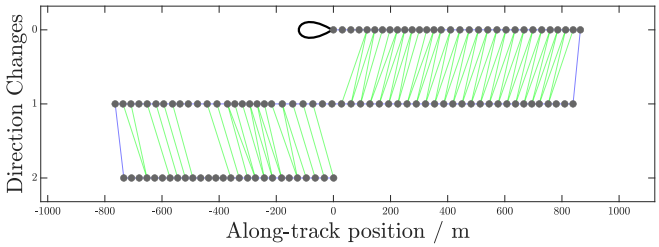

Figure 6. Structure of the pose-graph after optimization. Nodes are shown as gray dots, odometer edges are indicated by a blue line and loop closures by a green line. The prior edge is shows as a black loop at the first node.

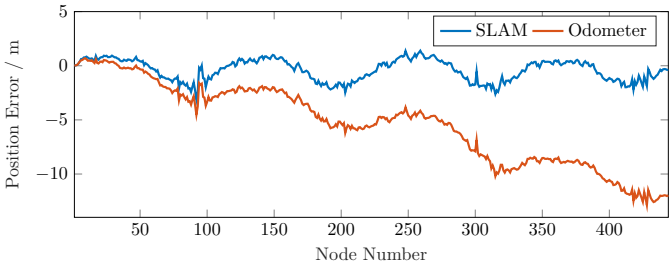

Figure 7. Position errors of the nodes after all four runs are processed. The odometer error keeps growing over time while the SLAM algorithm can bound the error and keep it in the range of a few meters. Note, the higher the node number, the later the node was created.

travels to Grunewald, where it changes its driving direction for the first time. From Grunewald, it then travels past the starting point to Halensee, where it changes direction for the second time. The graph shows that during the run a couple of loop closures are detected and inserted into the graph. These loop closures create constraints between nodes created close to the beginning, when the odometry only has accumulated a small error, and nodes that are created at a later stage where the odometry already accumulated a larger error. In the optimization, these constraints are then used to find a configuration of the nodes such that the cost function (9) is minimized.

From the results in Table II it becomes obvious that the proposed SLAM algorithm can improve the position error also over longer time intervals. While the odometry keeps drifting away from the true position, the SLAM solution bounds the error and achieves almost constant error statistics independent from the number of processed runs. This can be seen also in Fig. 7 where the position errors of the nodes are shown for the SLAM algorithm and the unaided odometry after processing all four runs. As for a single run, the errors are bounded here by loop closures. Within the four runs over 2000 loop closures are detected creating a rigid graph. This seems to be a lot compared to the low number of loop closures in Fig. 6 after one run. But here we have to consider that with each run we get more nodes for which a new node can detect a loop closure. Therefore, with more runs the number of loop closures and number of correlations that have to be calculated increases quickly. For online processing, this can

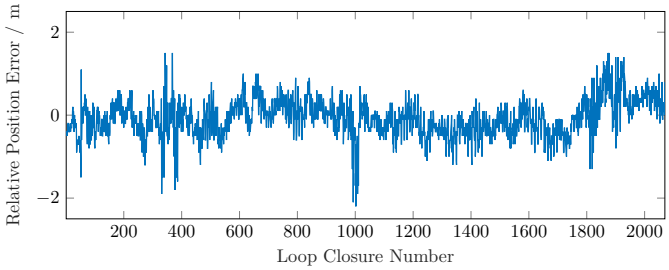

Figure 8. Error in the relative position for each loop closure detected during the four runs. The error is calculated based on the estimate obtained from the correlation and the GNSS ground truth.

cause problems because the computation time also increases. For the evaluation in this paper this was not yet a problem, i.e., to compute the complete SLAM algorithm for all four runs took less than 30 sec on a laptop processor. For comparison, the time the train was moving during these four runs was above 2000 sec. Nevertheless, for longer data sets it might be necessary to constrain the number of correlations or to use a faster correlation method, such as the one proposed in [12].

To also get an impression of how well the correlation can estimate the relative position of two nodes, the relative position error of the loop closures found during the four runs is shown in Fig. 8. It can be seen that the error is rather small and most of the time well below 2 m and has an RMSE of only $0.45\,\mathrm{m}$. Thus, the loop closure detection seems to be accurate and robust without any wrong loop closures.

VI. Conclusion

In this paper, we adapted the idea of graph SLAM to magnetic field-based localization in the railway domain. The key idea of the proposed SLAM algorithm is to use magnetic field distortions along a railway track to detect loop closures between two nodes in a pose-graph. For the detection, each node has its own dense local magnetic field map that represents the magnetic field around the nodes position. The dense map representation does not only enable the detection of loop closures but also the calculation of the relative position of two nodes. By adding the loop closures and the corresponding relative positions to the graph, the positions of the nodes are constrained and can be optimized to reduce their position errors.

To show the feasibility of the proposed algorithm, an evaluation was performed based on measurements recorded with the advanced TrainLab of the Deutsche Bahn while it was driving on a track in Berlin. The results show that SLAM with magnetic loop closure detection can bound the position errors of an odometry also for longer distances without requiring any prior knowledge of the track or the magnetic field.

Acknowledgment

This work was supported by the Ubiquitous Spatio-Temporal Learning for Future Mobility (ULearn4Mobility) Project funded by the Helmholtz AI Cooperation Unit under Grant ZT-I-PF-5-49. Furthermore, the authors would like to express their gratitude to the DB advanced TrainLab team for enabling the measurement campaign.

References

[1] B. Siebler, O. Heirich, S. Sand, and U. D. Hanebeck, "Joint Train Localization and Track Identification based on Earth Magnetic Field Distortions," in *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, Apr. 2020, pp. 941–948.

[2] R. Smith, M. Self, and P. Cheeseman, "Estimating Uncertain Spatial Relationships in Robotics," in *Autonomous Robot Vehicles*. New York: Springer, 1990, pp. 167–193.

[3] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: a factored solution to the simultaneous localization and mapping problem," in *Eighteenth National Conference on Artificial Intelligence*, USA, 2002, p. 593–598.

[4] S. Thrun and M. Montemerlo, "The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[5] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A Tutorial on Graph-Based SLAM," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[6] M. Kok and A. Solin, "Scalable Magnetic Field SLAM in 3D Using Gaussian Process Maps," in *2018 21st International Conference on Information Fusion (FUSION)*, 2018, pp. 1353–1360.

[7] I. Vallivaara, J. Haverinen, A. Kemppainen, and J. Röning, "Simultaneous localization and mapping using ambient magnetic field," in *2010 IEEE Conference on Multisensor Fusion and Integration*, 2010, pp. 14–19.

[8] E. Mendes, P. Koch, and S. Lacroix, "ICP-based pose-graph SLAM," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2016, pp. 195–200.

[9] J. H. Klüssendorff, J. Hartmann, D. Forouher, and E. Maehle, "Graph-based visual SLAM and visual odometry using an RGB-D camera," in *9th International Workshop on Robot Motion and Control*, 2013, pp. 288–293.

[10] J. Jung, J. Choi, T. Oh, and H. Myung, "Indoor Magnetic Pose Graph SLAM with Robust Back-End," in *Robot Intelligence Technology and Applications 5*. Springer, 2019, pp. 153–163.

[11] S. Sand, P. Unterhuber, D. B. Ahmed, F. De Ponte Müller, A. Lehner, I. Rashdan, M. Schmidhammer, R. Karásek, B. Siebler, O. Heirich, C. Gentner, M. Walter, S. Kaiser, M. Ulmschneider, M. Schaab, L. Wientgens, and T. Strang, "Radio interference measurements for urban cooperative intelligent transportation systems," in *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, 2021, pp. 1–6.

[12] K. Briechle and U. D. Hanebeck, "Template matching using fast normalized cross correlation," in *Optical Pattern Recognition XII*, 2001, pp. 95–102.