Kok, Manon; Solin, Arno

Scalable Magnetic Field SLAM in 3D Using Gaussian Process Maps

Published in:
21st International Conference on Information Fusion

DOI:
10.23919/ICIF.2018.8455789

Published: 05/09/2018

Please cite the original version:
Scalable Magnetic Field SLAM in 3D Using Gaussian Process Maps

Manon Kok*  
Delft University of Technology  
Email: m.kok-1@tudelft.nl

Arno Solin*  
Aalto University  
Email: arno.solin@aalto.fi

Abstract—We present a method for scalable and fully 3D magnetic field simultaneous localisation and mapping (SLAM) using local anomalies in the magnetic field as a source of position information. These anomalies are due to the presence of ferromagnetic material in the structure of buildings and in objects such as furniture. We represent the magnetic field map using a Gaussian process model and take well-known physical properties of the magnetic field into account. We build local maps using three-dimensional hexagonal block tiling. To make our approach computationally tractable we use reduced-rank Gaussian process regression in combination with a Rao–Blackwellised particle filter. We show that it is possible to obtain accurate position and orientation estimates using measurements from a smartphone, and that our approach provides a scalable magnetic field SLAM algorithm in terms of both computational complexity and map storage.

I. INTRODUCTION

The use of the magnetic field as a source of position information for indoor navigation is a promising novel approach that has gained interest in recent years [1]–[8]. It relies upon the spatial variation of the ambient magnetic field which is typically due to ferromagnetic material in the structures of buildings and to a lesser extent due to the presence of e.g. furniture. The advantage of using the magnetic field for positioning is that it can be measured by a small device, without additional infrastructure and without line-of-sight requirements. Furthermore, magnetometers are nowadays present in (almost) any smartphone. Crucial to this approach is the ability to build maps of the magnetic field that can be used for interpolation and extrapolation and the ability to localise inside this map. In this paper, we present a scalable and fully three-dimensional simultaneous localisation and mapping (SLAM) approach that builds a map of the magnetic field while simultaneously localising the sensor in the map.

Extending previous work [9], we build the magnetic field map using Gaussian process regression [10] and incorporate physical knowledge about the magnetic field known from Maxwell’s equations in the Gaussian process prior. To overcome issues with computational complexity, we use the rank-reduced approach first presented in [11]. This results in a representation that fits perfectly in a Rao–Blackwellised particle filter (RBPF) [12] which can be used both for building the map and for localisation in the map, resulting in a tractable localisation algorithm. Example results can be found in Fig. 1.

The approach presented in [11] relies on a basis function expansion on a specific domain where the required number of basis functions scales with the size of the domain. To allow for mapping large 3D areas, we propose a map representation using hexagonal block tiling, which aims at providing a very compact (in terms of required storage per volume) representation of the magnetic field map (all three vector field components and associated uncertainties). Using this approach, hexagonal block tiles are created whenever particles enter unexplored areas, resulting in a growing number of local Gaussian process maps.

We present experimental results where we obtain magnetometer measurements as well as odometry from a smartphone. This odometry gives us information about the change in position and orientation but the resulting pose estimates drift over time. This drift is corrected by making use of the magnetometer measurements and the magnetic field map. We show that accurate 6D pose and 3D map estimates can be obtained in large scale and 3D experiments, as illustrated in Fig. 1.

Previous work on magnetic field SLAM typically assumes that the motion is in 2D or 2.5D [13]–[16]. To the best of the authors’ knowledge, this is the first fully 3D approach to magnetic field SLAM. It is also the first time the reduced-rank approach from [11] has been used for SLAM and that the domain in this approach has been chosen to be hexagonal.

II. PROBLEM FORMULATION

We are interested both in estimating the position and the orientation of the sensor and in building a map of the magnetic

* This work was undertaken whilst MK was a Research Associate at the Department of Engineering, University of Cambridge, and AS was a Visiting Research Fellow at the Department of Engineering, University of Cambridge.
The dynamics of the position and orientation are modelled by \( \mathbf{p}_t^w \) and \( \mathbf{q}_t^{wb} \). The superscript \( w \) refers to the world frame. In our SLAM formulation, it is always possible to move or rotate the estimated map while simultaneously moving and rotating the position and orientation estimates. In other words, the absolute position and orientation are not observable. Because of this, without loss of generality we choose the origin of the world frame to be equal to the initial position of the sensor and its axes to be aligned with the coordinate system defined by the initial orientation.

The orientation of the sensor is denoted by \( \mathbf{q}_t^{wb} \), where \( b \) refers to the body frame. Its origin lies in the centre of the magnetometer triad and its axes are aligned with the axes of the magnetometer. We encode the orientation using a unit quaternion. There is a direct mapping from the quaternion \( \mathbf{q}_t^{wb} \) to the rotation matrix \( \mathbf{R}_t^{wb} \) and because of this we use these interchangeably. The double superscript \( wb \) denotes the rotation from body frame to world frame. The reverse rotation is denoted by \( \mathbf{b} \) and hence, \( \mathbf{R}_t^{wb} = (\mathbf{R}_t^{wb})^\top \).

We denote our estimate of the magnetic field map by \( \mathbf{m}_t \). More details about the exact form of \( \mathbf{m}_t \) can be found in Section III-A. Note that we assume that the map of the magnetic field at previously unseen locations. In practice, however, this quickly becomes computationally intractable because of the large amount of magnetometer measurements we collect and the fact that the computational complexity scales as \( \mathcal{O}(N_T^2) \). A computationally tractable solution was presented in [9] where the problem was projected on the eigenbasis of the (negative) Laplace operator in a confined domain \( \Omega \subset \mathbb{R}^3 \). In this domain, together with a prior on the state at the first time instance, the models (2) and (3) constitute the state space model that we use in the RBPF to estimate the state defined in (1).

### III. Methods

#### A. Gaussian process magnetic field map

Building on previous work [9], we build a Gaussian process magnetic field map in which we encode physical knowledge about the magnetic field. First of all, we know that the measured magnetic field consists of the Earth’s magnetic field and the magnetic field due to anomalies induced by small-scale variations and building structures. Furthermore, using Maxwell’s equations and classical electromagnetism [18], we know that the magnetic field due to the anomalies can be modelled using a latent scalar potential field \( \varphi(\mathbf{p}) \). Here, \( \varphi : \mathbb{R}^3 \to \mathbb{R} \), where \( \mathbf{p} \in \mathbb{R}^3 \) is the spatial coordinate in world frame and the magnetometer measures the derivative of this scalar potential. We assume the scalar potential field to be a realisation of a Gaussian process prior and the magnetometer measurements to be corrupted by Gaussian noise, resulting in the following model

\[
\varphi(\mathbf{p}) \sim \text{GP}(0, \kappa_{\text{lin}}(\mathbf{p}, \mathbf{p}') + \kappa_{\text{SE}}(\mathbf{p}, \mathbf{p}')), \\
y_{m,t} = \mathbf{R}_t^{bw} \nabla \varphi(\mathbf{p})|_{\mathbf{p} = \mathbf{p}_t^w} + \epsilon_{m,t},
\]

where \( \epsilon_{m,t} \sim \mathcal{N}(0, \sigma_{\text{m}}^2 \mathbf{I}_3) \), for each observation \( t = 1, 2, \ldots, N_T \). The local Earth’s magnetic field contributes linearly to the scalar potential which we model in terms of a linear covariance function [19] as

\[
\kappa_{\text{lin}}(\mathbf{p}, \mathbf{p}') = \sigma_{\text{lin}}^2 \mathbf{p} \mathbf{p}'^\top,
\]

where \( \sigma_{\text{lin}}^2 \) is the magnitude scale hyperparameter. The magnetic field anomalies are captured by the squared exponential covariance function

\[
\kappa_{\text{SE}}(\mathbf{p}, \mathbf{p}') = \sigma_{\text{SE}}^2 \exp\left(-\frac{\|\mathbf{p} - \mathbf{p}'\|^2}{2\ell^2}\right),
\]

where \( \sigma_{\text{SE}}^2 \) is a magnitude and \( \ell \) the characteristic length-scale hyperparameter. Including the additional prior information introduced by the scalar potential in (4) results in a coupling between the three components of the magnetic field according to physical laws. This has been shown in [9] to improve prediction accuracy over simply modelling each magnetic field component separately.

Using the model (4), it is possible to predict the magnetic field at previously unseen locations. In practice, however, this quickly becomes computationally intractable because of the large amount of magnetometer measurements we collect and the fact that the computational complexity scales as \( \mathcal{O}(N_T^2) \).
the eigendecomposition of the Laplace operator subject to Dirichlet boundary conditions can be solved as
\[
\begin{aligned}
-\nabla^2 \phi_j(p) &= \lambda_j^2 \phi_j(p), \quad p \in \Omega, \\
\phi_j(p) &= 0, \quad p \in \partial \Omega.
\end{aligned}
\]
(7)

Using this eigendecomposition, it is possible to approximate the covariance function in (4) as
\[
\kappa(p, p') = \kappa_{\text{lin.}}(p, p') + \kappa_{\text{SE}}(p, p') \\
\approx \kappa_{\text{lin.}}(p, p') + \sum_{j=1}^{m} S_{\text{SE}}(\lambda_j) \phi_j(p) \phi_j(p'),
\]
(8)

where \(\kappa_{\text{SE}}(p, p')\) is approximated using \(m\) basis functions \(\phi_j\) and their corresponding eigenvalues \(\lambda_j\). The exact form of \(\phi_j\) and \(\lambda_j\) depends on the shape of the domain \(\Omega\). Note that \(S_{\text{SE}}(\cdot)\) denotes the spectral density function of the squared exponential kernel \(S_{\text{SE}}(\cdot)\) for which a closed form solution exists, see [9].

Using the approximation (8) with the boundary conditions defined in (7) implies that our model of the magnetic field reverts back to the Earth’s magnetic field at the boundary of the domain.

A fixed domain \(\Omega\) is not desirable in SLAM, as we do not know \textit{a priori} the spatial extent of the final magnetic field map. Furthermore, the number of basis functions that is needed for a good approximation scales with the size of the domain and for large scale SLAM problems will hence become intractable. We therefore propose a scalable representation where we split the magnetic map into a three-dimensional grid of hexagonal blocks with radius \(r\) and height \(2L_z\), such that each subdomain \(d\) is given by
\[
\Omega^{(d)} = \{ p \mid (p_1, p_2) \in \text{hexagon}(r, p^{(d)}_{1,2}), \\
p_3 \in [p^{(d)}_3 - L_z, p^{(d)}_3 + L_z] \}. \tag{9}
\]

Here, \(p^{(d)}\) is the center point of the \(d\)th hexagon cell in the grid representation. The hexagonal grid provides an efficient way of maximising the area per number of basis functions for representing the magnetic field map, thus requiring less memory for storing the map representation.

To compute the basis functions \(\phi_j\) and their corresponding eigenvalues \(\lambda_j\), we need to solve the eigendecomposition of the Laplace operator (7) in a two-dimensional hexagon. This can not be solved in closed form, but instead it can be solved numerically. We set up the eigenvalue problem by composing the sparse stencil matrix corresponding to the Laplacian using a nine-point finite difference approximation. The solution itself is given by the Lanczos algorithm (see, \textit{e.g.}, [20], callable in Matlab through \texttt{eigs}) that we use for solving the \(m\) largest real eigenvalues \(\lambda_{\text{hex}}^m\) together with the corresponding eigenfunctions \(\phi_{\text{hex}}^m(p)\). Fig. 2 shows the 40 first eigenfunctions of the negative Laplacian in a unit hexagonal domain with respect to Dirichlet boundary conditions.

The eigendecomposition is extended to cover the three-dimensional hexagonal grid cell \(\Omega^{(d)}\) by considering the numerically solved horizontal eigendecomposition and a closed-form solution to the separable vertical dimension. This choice is motivated by the assumption that paths typically exhibit most variation in the horizontal direction while exhibiting less variation in the vertical direction. In other words, we assume most variation to happen in horizontal cross sections since human motion tends to be along two-dimensional planes with less-frequent ascending and descending. None of the computational benefits rely on this assumption, and it is also a sensible choice for other (non-bipedal) movement.

The final eigendecomposition then becomes the combination:
\[
\phi_j(p) = \phi_{\text{hex}}^{n_{j,1}}(p_{1,2}) \left[ \frac{1}{\sqrt{L_z}} \sin \left( \frac{\pi n_{j,1}(p_3 + L_z)}{2L_z} \right) \right], \tag{10}
\]
\[
\lambda_j^2 = \left[ \lambda_{\text{hex}}^{n_{j,1}} \right]^2 + \left( \frac{\pi n_{j,2}}{2L_d} \right)^2, \tag{11}
\]

where the matrix \(n \in \mathbb{R}^{m \times 2}\) consists of an index set of permutations of integers \(\{1, 2, \ldots, m\}\) (\textit{i.e.}, \(\{1, 1\}, \{2, 1\}, \ldots, \{1, 2\}, \ldots, \{2, 2\}\)) corresponding to the \(m\) largest eigenvalues.

The numerical solution for the two-dimensional hexagon is fast and takes less than a second even with a tight discretisation on an average laptop computer. Furthermore, because the basis functions (10) are independent of the hyperparameters and the input locations, they only need to be evaluated once, not causing any computational overhead in our SLAM algorithm.

The basis function expansion described above allows us to write the prior model for each hexagonal tile at time \(t = 0\) in terms of a mean \(\mathbf{m}_0 = \mathbf{0}_{m+3}\) and a covariance
\[
\mathbf{P}_0 = \text{diag} \left( \sigma_{\text{lin.}}^2, \sigma_{\text{lin.}}^2, S_{\text{SE}}(\lambda_1), S_{\text{SE}}(\lambda_2), \ldots, S_{\text{SE}}(\lambda_m) \right). \tag{12}
\]

The posterior can be updated sequentially [21] as new magnetometer data arrives at \(t = 1, 2, \ldots, N_T\) and can be written as
\[
\begin{aligned}
S_t &= C_t S_{t-1} C_t^T + \sigma_m^2 \mathbf{I}_3, \\
K_t &= P_{t-1} C_t^T S_{t-1}^{-1}, \\
m_t &= m_{t-1} + K_t (y_{m,t} - C_t m_{t-1}), \\
P_t &= P_{t-1} - K_t S_t K_t^T.
\end{aligned}
\]
(13)

with \(C_t = R_{bw} \nabla \Phi_t\) and
\[
\nabla \Phi_t = \left( \nabla p_t^T, \nabla \phi_1(p_t), \ldots, \nabla \phi_{m}(p_t) \right). \tag{14}
\]
Note that (13) is a Kalman filter measurement update of the state $\mathbf{m}$, representing the magnetic field map in a specific tile. This fact will be exploited in our magnetic field SLAM algorithm presented in Section III-B.

### B. Magnetic field SLAM

As discussed in Section II, we are interested in estimating the position and the orientation of the sensor while building a map of the magnetic field. We represent the map in terms of a varying number of hexagonal block tiles, each using the basis function expansion presented in Section III-A. We use a nonlinear filtering technique to estimate the position, the orientation and the map of each tile that has been visited. The state vector (1) has a relatively large dimension due to the representation of the magnetic field map. We represent the map in terms of the position and the orientation of the sensor while building a map of the magnetic field. We initialise the magnetic field according to

$$ q_0^{\text{wb}} = \mathbf{wb}_0, \quad \{\mathbf{m}_i\}_{d=1}^{N_D,t} = \{\mathbf{m}_i\}_{d=1}^{N_D,t}. $$

Here, $N_D,t$ is the number of hexagonal block tiles that have been created up to time $t$.

In the RBPF, $i = 1, \ldots, N_p$ particles having a position $p_i^{w(i)}$ and an orientation $q_i^{wb(i)}$ are used to represent the state at time $t$.

Each of the particles also consists of the map $\mathbf{m}_i(t)$ and its covariance $P_i(t)$ of the magnetic field map of each hexagonal block that particle $i$ has visited. We assume that each tile is independent. The covariance of the linear state can therefore be described as

$$ \mathbf{P}_t = \{\mathbf{P}_i\}_{d=1}^{N_D,t}. $$


The dynamics of the state is chosen as in (2) and the measurement model is given in (3). It is important to note that, as discussed in Section II, the absolute position and orientation are not observable. Because of this, we use the translation and rotation $p_0^{w(i)}$ and $q_0^{wb(i)}$ of all particles $i = 1, \ldots, N_p$ equal to an initial position and orientation $p_0^{w(0)} = 0_3$, $q_0^{wb(0)} = q_0^{wb}$. We initialise the magnetic field according to the Gaussian process prior (see also Section III-A).

The state space model presented in Section II results in a fairly straightforward RBPF implementation. The major difference from a standard RBPF is that each particle only updates the local magnetic field map at its current location. Note that the computational complexity of the algorithm scales linearly with the number of particles and linearly with the number of hexagonal tiles per particle. The algorithm is summarised in Alg. 1.

In a practical magnetic field SLAM implementation care must be taken of the following in terms of updating the map:

- To avoid boundary effects due to the Dirichlet boundary conditions, the actual tile extends slightly outside of the hexagonal block domain for updating.

- To allow for a smooth transition from one tile to the next, particles close to the border of a tile also update the map of the neighbouring tiles.

During the “exploration phase” when particles start building a map of previously unseen locations, very little information is available to distinguish between the particles. Because of this, we would like to ensure that the particle cloud keeps spreading during this phase. To this end, we adapt the RBPF implementation in two ways:

- To maintain the particle spread during the exploration phase, we delay the updating of the map by one timescale $\ell$.

- To not collapse the particle cloud too quickly, we only resample the particles when they revisit a tile. Resampling is further delayed until at least 90% of the particles has arrived in a tile that they revisit.

### IV. RESULTS

In the empirical experiments, we use the built-in magnetometer and odometry data captured by a smartphone for simultaneously building the magnetic field map and localising on it.
We use an Apple iPhone 6s (published in September 2015), a previous flagship product of Apple which by current standards represents more of a standard smartphone.

The interest in this paper is purely on building of the map and on localisation, so we choose to leverage on the recent developments in visual-inertial pedestrian dead-reckoning (PDR) methods and use the built-in ARKit PDR provided by Apple. ARKit (released September 2017, used version running in iOS 11.2.5) is using the IMU and camera of the phone to provide a 6-DoF (position and orientation) PDR movement trajectory. ARKit does visual relocalisation under the hood, which aims to correct for long-term drift, resulting in discontinuities in the PDR track. These discontinuities turned out to be more harmful than useful for our approach, and we implemented a heuristic filtering approach for removing the relocalisation jumps. The PDR—albeit being surprisingly robust—suffers from long-term drift both in position and orientation (see Fig. 3).

The data was captured by a customised data capture application running on the phone. It collects the IMU (at 100 Hz) and three-axis magnetometer (100 Hz) data time-locked to the ARKit PDR estimates (60 Hz). Furthermore, the captured data was complemented with video frames of the back-facing phone camera (resolution 1280×720 at 60 fps), the barometric air pressure (10 Hz), and GNSS locations captured by the phone, none of which were used in the SLAM implementation, but captured for validation and reference. All data was stored on the device, downsampled to 10 Hz before use, and the SLAM implementation was run off-line in a Mathworks MATLAB implementation.

For simplicity and generalisability, we use the same model parameters in all our experiments. We choose the settings for the RBPF and the hexagonal blocks to be:

- We set the size of the hexagonal blocks to $r = 5$ m, $L_z = 2$ m and the number of basis functions per tile to $m = 256$. The radius and height used in solving the eigenbasis of the Laplacian extends one meter outside the hexagon.
- The number of particles in the RBPF is $N_P = 100$.
- Neighbouring tiles are updated if samples are closer than 10 cm to the hexagonal block boundary.
- The process noise parameters are set to $\Sigma_p = \text{diag}(0.1^2, 0.1^2, 0.02^2)$ (values specified in meters drift per second) and $\Sigma_q = \text{diag}(0.01^2, 0.01^2, 0.24^2)$ (values specified in degrees drift per second).
In terms of memory usage, these choices mean that for each initialised map tile, the particle state dimension grows by 256. For fast look-up, our implementation uses a hashmap (per particle) for storing and keeping track of the map tiles. Each tile stores the GP mean and covariance matrix.

We use the same Gaussian process hyperparameter values for each of the experiments. These hyperparameters are typically learned from data. However, in sequential algorithms, hyperparameter values need to be available already before any data is processed. We therefore choose values that seem physically reasonable and that fit the data in the different scenarios that we consider reasonably well. We set the Gaussian process hyperparameters defined in Section III-A to \( \sigma_n^2 = 650, \sigma_{\mu}^2 = 200, \ell = 1.3, \) and \( \sigma_{\mu}^2 = 10 \) (the length-scale units are meters, the magnitude parameters in \( \mu \) T). This assumes that the lengthscale of the magnetic field anomalies is in the order of 1.3 m. The measurement noise standard deviation \( \sigma_n \) is rather high to also account for model discrepancies.

A. Illustrating magnetic field SLAM in 2D

In a first experiment, a data set is collected in the Design & Project Office (DPO) at the Engineering Department of the University of Cambridge. Significant excitation in the magnetic field is present for instance due the presence of radiators and a large number of computers. The trajectory is around 250 meters long and covers an area of around \( 20 \times 5 \) m. The same circular trajectory is traversed several times. Because of this, it is easy to visually inspect the quality of the estimates from our SLAM algorithm which are shown in Fig. 3. The different subfigures illustrate the workings of our algorithm. As a supplementary file to this paper, there is also a video\(^1\) demonstrating the workings of the SLAM algorithm.

Figs. 3(a) – 3(f) show how the estimates progress over time. For visualisation purposes, the norm of the magnetic field is predicted by the map of the highest weight particle at a large number of locations in each hexagon. The transparency in the map visualises the uncertainty of the map. It can be seen that the uncertainty is large in unexplored regions (no map is actually visible) and that the uncertainty of the map decreases over time in the re-visited regions.

The black line displays the trajectory of the highest weight particle. In Figs. 3(a) and (b), the deliberate slight delay in updating the map can clearly be seen. As discussed in Section III-B, we wait with resampling until at least 90% of the particles is in a tile that they have previously updated and left. Because of this, no resampling is done in the first loop, until around 50 seconds and the spread in the particles in Fig. 3(c) is therefore fairly large. This allows the algorithm to properly close the loop as visualised in Fig. 3(d). Another aspect to note about our algorithm is that in Fig. 3(c), most particles are on the border between the tile in which the trajectory was started and a new tile below. To allow for smooth transitions between these two tiles, many of the particles will in this case update the local magnetic field maps in both tiles.

The map that is visualised in Figs. 3(a) – (f), shows the norm of the predicted magnetic field. Our model (4), however, explicitly models the magnetic field as a three-dimensional vector. Instead of visualising the norm of the predicted magnetic field, it is therefore also possible to visualise the predicted magnetic field in three directions. This is shown in Figs. 3(g) – (i). Because the dip angle of the Earth’s magnetic field is fairly large in Cambridge, UK, the \( z \)-component of the magnetic field is of significantly larger magnitude than the \( x \)- and \( y \)-components.

Based on Fig. 3(f) it can be concluded that our SLAM algorithm estimates the trajectory with high accuracy since the same trajectory can be seen to be traversed multiple times. For comparison, we also show the odometry in Fig. 3(j) which clearly shows a drift in the position. Our magnetic SLAM approach is able to correct for this drift. To illustrate the fact that even though the movement for this data set is in a 2D plane the estimation is still done in 3D, the tilted 3D view is shown in Fig. 3(k).

B. Fully 3D magnetic field SLAM

In Section IV-A, the movement was close to two-dimensional. However, exactly the same procedure also applies to movement and block tile structures in three dimensions. In Fig. 4 we show the estimated trajectory and the magnetic field map for an empirical experiment in which we first walked up and subsequently down a set of stairs at the Engineering Department of the University of Cambridge.

Again, each hexagonal block has its own local magnetic field map. To visualise these, the norm of the predicted magnetic field for a hexagonal slice in the middle of each block is shown. The transparency visualises the uncertainty of

---

\(^1\)The supplementary video is available on YouTube: [https://youtu.be/pbwWLoh6mvI](https://youtu.be/pbwWLoh6mvI)
the odometry we obtain from the ARKit does not always obey our motion model. More specifically, for straight paths the odometry shows almost no drift, while fast turns can introduce sudden errors. This does not follow our motion model in which we model a constant process noise. In future work we would like to focus on using inertial PDR as odometry instead. Apart from avoiding these issues, this would also result in a positioning algorithm only based on widely available inertial and magnetometer sensors. Other directions of future work are in the direction of using a Rao-Blackwellised Particle Smoother and further reducing the computational complexity of the algorithm.

ACKNOWLEDGMENTS

This research was financially supported by the EPSRC grant Autonomous behaviour and learning in an uncertain world (Grant number: EP/J012300/1) and the Academy of Finland grant Sequential inference for real-time probabilistic modelling (Grant number: 308640).

REFERENCES


