Moving Horizon Estimation SLAM for agile vehicles in 3-D environments

Daniel Sousa[1] and Bruno J. Guerreiro[1,2]

[1] NOVA School of Science and Technology, UNINOVA-CTS, LASI, NOVA University Lisbon, Caparica, Portugal
[2] Institute for Systems and Robotics (ISR/LARSYS), Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal
E-mails: `dmd.sousa@campus.fct.unl.pt, bj.guerreiro@fct.unl.pt`

Abstract. The ability for a robot to be able to construct a map of the environment and recognize its position on it was one of the biggest developments in robotics. Simultaneous localization and mapping (SLAM) framework builds onto the perception of the robot, giving it the possibility to online calculate its trajectory and avoid obstacles. Moreover, the continuous development of processing units has given the possibility for previously hardware exhausting solutions to be considered as an option for the localization and mapping problem. With this in mind, this work is focused on developing a SLAM solution for a 6 DoF vehicle operating on a 3-D environment using moving horizon estimation (MHE). Throughout the paper it is tested the applicability of the proposed solution in a simulation environment of two loop square-shaped corridors with stationary landmarks, whilst comparing the obtained results with another probabilistic approach, the EKF, which is commonly used but loses stability on extremely nonlinear dynamics. Each of the algorithms is simulated in MATLAB.

Keywords: SLAM, MHE, UAV, EKF

1 Introduction

Simultaneous localization and mapping (SLAM) is a well-known robotics problem for, as the name implies, the simultaneous effort of mapping a not previously known environment and localizing the robot on the constructed map. The core of the problem, firstly introduced in [4], is still a relevant issue with no definite solution, as there have been a continuous research, for over three decades, on how to solve it considering distinct types of robot applications and environments. This is specially the case for vehicles with nonlinear system dynamics, like satellites and unmanned aerial vehicles (UAVs), which in some cases can be deployed in urban 3-D environments, and are characterized by having nonlinear motion and observation models.

There are several distinct algorithms to solve the SLAM problem of vehicles with non-linear dynamics. The better-known are the Kalman filter (KF) and its nonlinear versions, such as the extended Kalman filter (EKF) and unscented Kalman filter (UKF), but also the particle filter and graph-based algorithms.

The Moving horizon estimation (MHE) is defined as a constrained, nonlinear optimization problem [16], that unlike the KF and EKF, does not assume the noise to be Gaussian [18]. Similarly to the Full Information filter, the MHE has in consideration the measured data at the current and previous steps. However, it uses a small window of measured data to make its' estimations, which makes it a more efficient and less exhaustive estimator as the Full Information filter considers every past estimation. Changing the size of the horizon creates a compromise between the computational requirements and the performance of the method. In spite of that, the MHE still has a high computational cost [17]. Moreover, the stability of the MHE for linear and nonlinear systems has been studied in [12], [15] and [14], where it is showed that this technique is an asymptotically stable observer in a nonlinear deterministic modeling framework. In addition to that, [13] found that the MHE for nonlinear detectable systems with bounded disturbances is robust and GAS. There is still a limited number of work done regarding the use of MHE on SLAM. Moreover, there are not many applications of this technique on a vehicle with complex dynamics, such as vehicles with 6 DoF. In [10] it's considered the effectiveness of a MHE approach to the SLAM of a tricycle in a crowded environment, and in [11] the MHE is applied on multi-robot SLAM. Other relevant applications include hybrid systems [6] or distributed estimators [5].

This work addresses the performance and viability of a SLAM algorithm based on moving horizon estimation (MHE) applied to a rigid body, whilst having in consideration the high needs of computational power that the algorithm requires. The advantage of this method is its applicability on nonlinear systems, since it does not have the linearity constraints imposed by other commonly used solutions, such as the extended Kalman filter (EKF), which needs a local linearization. The performance of the proposed solution will then be compared to a SLAM based EKF approach.

The remainder of this paper presents the system motion dynamics in Section 2, and the MHE filter in Section 3. A comparison of the proposed algorithm with the EKF is given in Section 4. Final remarks and future work directions are provided in Section 5.

2 System Dynamics

The system dynamics was designed in order to model the behavior of any vehicle in 3-D space, as it considers it as the movement of an abstract rigid body.

2.1 Notation

In order to better comprehend the mathematical expressions throughout this document some abbreviations were taken into account. Every vector is written in a bold lower case and each matrix on a bold higher case, like $\boldsymbol{v}$ and $\boldsymbol{M}$, respectively, whilst the scalar values are represented by plain letters. $\mathbb{H}$ defines the quaternions group, and $\mathbb{H}_p$ the group of pure quaternions $\mathbb{H}_p = \{\boldsymbol{q} \in \mathbb{H} \mid \boldsymbol{q} = (0, \boldsymbol{q}_v)\}, \boldsymbol{q}_v \in \mathbb{R}^3$. The sympletic group, $Sp(1)$, is the orthogonal group formed by the unitary quaternions, which is isomorphic to the unit sphere in $\mathbb{R}^4$, $S^3 = \{\boldsymbol{q} \in \mathbb{H} \mid |\boldsymbol{q}|= 1\}$ [7]. The special orthogonal group in 3-D space is defined as $SO(3) := \{\boldsymbol{R} \in \mathbb{R}^{3\times 3} \mid \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}_3, det(\boldsymbol{R}) = 1\}$, where $(.)^T$ is the transpose superscript. $\boldsymbol{S}[.]$ is a map from $\mathbb{R}^3$ to the space of three-by-three skew-symmetric matrices, $so(3)$, defined as

$$\boldsymbol{S}[\boldsymbol{a}] = \begin{bmatrix} 0 & -a_1 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}, \boldsymbol{a} = \begin{bmatrix} a_1 \; a_2 \; a_3 \end{bmatrix}^T \in \mathbb{R}^3 \tag{1}$$

such that $\boldsymbol{S}[\boldsymbol{a}]\boldsymbol{u} = -\boldsymbol{S}[\boldsymbol{u}]\boldsymbol{a}$, whereas $\boldsymbol{S'}[.]$ is given by

$$\boldsymbol{S'}[\boldsymbol{a}] = \begin{bmatrix} 0 & -a_1 & -a_2 & -a_3 \\ a_1 & 0 & a_3 & -a_2 \\ a_2 & -q_3 & 0 & a_1 \\ a_3 & a_2 & -a_1 & 0 \end{bmatrix}. \tag{2}$$

Also consider that ${}^W_B\boldsymbol{R} \in SO(3)$, or simply $\boldsymbol{R}$, is the rotation matrix from B, the Body frame, to W, the World frame, whereas the position of origin of the body frame described in the world frame is simply denoted as $\boldsymbol{p} \in \mathbb{R}^3$, which can be defined as a vector with x, y, and z coordinates.

2.2 Motion Model

For the purpose of the motion model it was opted to view it as a simple rigid body. This decision enables the use of this filter to any type of vehicle, as long as the velocities at play on the rigid body are available. The proposed model works by considering the measurements of the vehicle's linear and angular velocities applied to the body, $\boldsymbol{v}_m(t)$ and $\boldsymbol{\omega}_m(t)$ respectively, which can be defined as the sum of the actual value and the noise associated with the environment, sensor and system, as follows

$$\boldsymbol{v}_m(t) = \boldsymbol{v}(t) + \boldsymbol{\xi}_v(t) \tag{3}$$

$$\boldsymbol{\omega}_m(t) = \boldsymbol{\omega}(t) + \boldsymbol{\xi}_\omega(t) \tag{4}$$

Therefore, the vehicle motion model can be defined as

$$\begin{cases} \dot{\boldsymbol{p}}(t) = \boldsymbol{R}(\boldsymbol{q}(t))\boldsymbol{v}_m(t) \\ \dot{\boldsymbol{q}}(t) = \boldsymbol{S'}[\boldsymbol{\omega}_m(t)]\boldsymbol{q}(t) \end{cases} \tag{5}$$

in which $\boldsymbol{R}(\boldsymbol{q}(t)) \in SO(3)$, is the transformation of the quaternion $\boldsymbol{q}$ to a rotation matrix and $\boldsymbol{S'}$ is a the skew-symmetric matrix defined in (2). In addition to the vehicle motion, it is assumed that the landmarks are static, resulting in the i^{th} landmark motion model

$$\dot{\boldsymbol{p}}_i(t) = \boldsymbol{0}. \tag{6}$$

The decision of defining the vehicles orientation by a quaternion representation over other options like the rotation matrices and Euler angles was derived by two factors. One of them was the number of variables in each option. While the quaternion is represented by a 4-by-1 vector, the 3-D rotation matrices are represented by a 3-by-3 matrix which has nine elements, instead of the four elements of the quaternion. This is relevant due to the number of variables in the vector state that the MHE would have to consider and estimate, which would demand a bigger work load on the algorithm that minimizes the cost function. As this method is known for its computational cost the most wise option is to make its life easier and choose the quaternion representation. The Euler angles, which are able to define the attitude of an object with only three variables, were also considered but as they have a singularity whenever the angle of the second rotation is equal to 90 degrees (or 270 degrees) they were not used.

Considering that within a sampling period Δ_t the linear and angular velocity measurements are constant or slowly variable, the discretization of the first equation of (5), representing the position kinematics, uses the forward Euler method [3], whereas the second equation, describing the rotation kinematics using unit quaternions, was discretized considering the exponential map, so that it is possible to guarantee that the updated quaternion is still a unit quaternion. Thus, the resulting vehicle and landmarks discretized motion model is given

$$\begin{cases} \boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \Delta_t \boldsymbol{R}(\boldsymbol{q}_k)\boldsymbol{v}_{mk} \\ \boldsymbol{q}_{k+1} = e^{\Delta_t \boldsymbol{S'}[\boldsymbol{\omega}_{mk}]}\boldsymbol{q}_k \\ \boldsymbol{p}_{ik+1} = \boldsymbol{p}_{ik} \ , \ \forall_{i \in I_L} \end{cases} \tag{7}$$

where the set $I_L = \{1, 2, \ldots, n_L\}$ accounts for all landmarks in the state vector. Consider also the state vector defined as $\boldsymbol{x}_k = \begin{bmatrix}\boldsymbol{x}_{Vk}^T \ \boldsymbol{x}_{Lk}^T\end{bmatrix}^T$, where $\boldsymbol{x}_{Vk} = \begin{bmatrix}\boldsymbol{p}_k^T \ \boldsymbol{q}_k^T\end{bmatrix}^T$ denotes the vehicle related states, and $\boldsymbol{x}_{Lk} = \begin{bmatrix}\boldsymbol{p}_{1k}^T \cdots \boldsymbol{p}_{n_L k}^T\end{bmatrix}^T$ the landmarks related states. Further considering the input vector $\boldsymbol{u}_k = \begin{bmatrix}\boldsymbol{v}_{mk}^T \ \boldsymbol{\omega}_{mk}^T\end{bmatrix}^T$, we can define the motion dynamics as $\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)$ or in more detail as

$$\begin{cases} \boldsymbol{x}_{Vk+1} = \boldsymbol{f}_V(\boldsymbol{x}_{Vk}, \boldsymbol{u}_k) \\ \boldsymbol{x}_{Lk+1} = \boldsymbol{f}_L(\boldsymbol{x}_{Lk}) \end{cases} \tag{8}$$

2.3 Observation Model

The observation model consists of the position of the observed landmarks, described in the frame of the vehicle's visual sensor, or for simplicity, in the body frame, ${}^{B}\boldsymbol{p}_{ik}$. As a result, the observation model can be defined for a given landmark i, by converting the landmark position from the world frame to the vehicle frame, using

$$\boldsymbol{y}_{ik} = \boldsymbol{h}_i(\boldsymbol{x}_{Vk}, \boldsymbol{p}_{ik}) := \boldsymbol{R}^T(\boldsymbol{q}_k)(\boldsymbol{p}_{ik} - \boldsymbol{p}_k) \ . \tag{9}$$

Further consider the partition of the landmark states into the set of observed or measured landmarks, I_O and the set of unobserved landmarks, I_U, such that

$I_L = I_O \cup I_U$. These sets can change in each iteration, and for simplicity are defined as $I_O = \{1, 2, \dots, n_O\}$ and $I_U = \{n_O + 1, \dots, n_L\}$, which can be used to further decompose the landmark state vector into $\boldsymbol{x}_{Lk} = \left[\boldsymbol{x}_{Ok}^T \; \boldsymbol{x}_{Uk}^T\right]^T$. Using this definitions, we can define the measurement vector as

$$\boldsymbol{y}_k := \left[\boldsymbol{y}_{1k}^T \; \boldsymbol{y}_{2k}^T \cdots \boldsymbol{y}_{n_O k}^T\right]^T = \boldsymbol{h}(\boldsymbol{x}_{Vk}, \boldsymbol{x}_{Ok}) \tag{10}$$

3 Moving horizon estimation SLAM filter

Conversely to the EKF, the MHE considers the previous robot states, $\boldsymbol{p}_k$, in each iteration, in a similar manner to the full information estimator, but with a restricted discrete-time horizon H of past samples. As such, the estimated state vector, $\hat{\boldsymbol{x}}_H$, of the MHE is given by $\hat{\boldsymbol{x}}_{Hk} = \left[\hat{\boldsymbol{x}}_{Vk}^T \cdots \hat{\boldsymbol{x}}_{Vk-H}^T \; \hat{\boldsymbol{x}}_O^T\right]^T$, which is comprised of vehicle states, $\hat{\boldsymbol{x}}_{Vk}$, and the observed landmarks positions, $\hat{\boldsymbol{x}}_O$, over the estimation horizon H. This definition assumes only one estimate for the observed landmarks, as they are static. The general structure of the MHE optimal estimation problem can be defined as the minimization of a stage cost for each horizon instant, l_k of the estimation errors associated with the vehicle motion model and landmark observation model, $\boldsymbol{w}_k$ and $\boldsymbol{n}_k$, respectively added to first equation of (8) and to (10), and the arrival cost l_{k_0} [9], as defined by

$$\begin{aligned} \min_{\boldsymbol{x}_{Hk_0}, \boldsymbol{w}_k, \boldsymbol{n}_k} \quad & J := l_{k_0}(\boldsymbol{x}_{Hk_0}) + \sum_{k=k_0-H}^{k_0} l_k(\boldsymbol{w}_k, \boldsymbol{n}_k) \\ s.t. \quad & \boldsymbol{x}_{Vk+1} = \boldsymbol{f}_V(\boldsymbol{x}_{Vk}, \boldsymbol{u}_k) + \boldsymbol{w}_k \;, \; \forall_{k=k_0-H,\dots,k_0} \\ & \boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_{Vk}, \boldsymbol{x}_{Ok}) + \boldsymbol{n}_k \;, \; \forall_{k=k_0-H,\dots,k_0} \end{aligned} \tag{11}$$

in which k_0 is the current SLAM time instant and the cost functions $l_{k_0}(.)$ and $l_k(.,.)$ are usually quadratic functions of their arguments. This minimization problem can be regarded as an unconstrained optimization problem with optimization variable $\boldsymbol{x}_{Hk_0}$ by redefining the cost function J as

$$\begin{aligned} J = \|\hat{\boldsymbol{x}}_{Hk_0} - \hat{\boldsymbol{x}}_{Hk_0}^-\|^2_{\boldsymbol{\Sigma}_k^{-1}} + \sum_{k=k_0-H}^{k_0} & \|\hat{\boldsymbol{x}}_{Vk+1} - \boldsymbol{f}_V(\hat{\boldsymbol{x}}_{Vk}, \boldsymbol{u}_k)\|^2_{\boldsymbol{Q}_k^{-1}} \\ & + \sum_{i \in I_{Ok}} \|(\boldsymbol{y}_{i,k} - \boldsymbol{h}_i(\hat{\boldsymbol{x}}_{Vk}, \hat{\boldsymbol{p}}_i))\|^2_{\boldsymbol{R}^{-1}}, \end{aligned} \tag{12}$$

where I_{O_k} is the set of observed landmarks at time k, $\hat{\boldsymbol{x}}_{Hk_0}$ is the current state vector estimation and $\hat{\boldsymbol{x}}_{Hk_0}^-$ the previous state vector estimation. The second term in equation (12) has in consideration the error of the vehicles motion dynamics, while the third term evaluates the error related to the measurements taken of each observed landmark. The first term is the arrival cost, which has in account the previous estimation.

Additionally, the weights $\boldsymbol{R}, \boldsymbol{Q}_k, \boldsymbol{\Sigma}_k$ are the measurement noise, process noise and system EKF covariances, respectively. These matrices will be calculated before each iteration of the MHE, as an EKF algorithm will run in series with it. This interaction is depicted in Figure 1, in which it is noted that both filters receive sensor and measurement data and propagate their own estimation, whilst the MHE waits for the covariance matrices of the EKF. The choice of designing a probabilistic MHE, thus using the covariances matrices as weights, was made having in account the robustness of this approach. The execution speed of the EKF is considerably faster than the MHE.

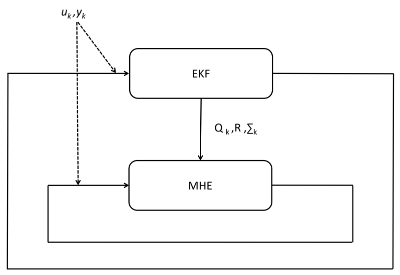
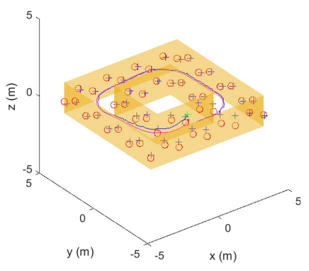


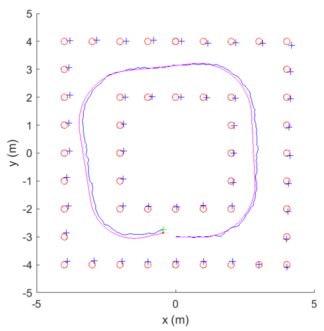
Fig. 1: MHE+EKF Algorithm

3.1 Simulation

Using the simulation environment shown in Figure 2 it was possible to test the effect of the window size and number of landmarks on the computing time. On Figure 2, the true landmark positions and trajectory are represented in purple, whilst their estimated values are in blue.



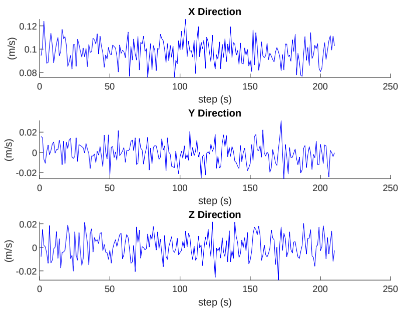
(a) Custom view



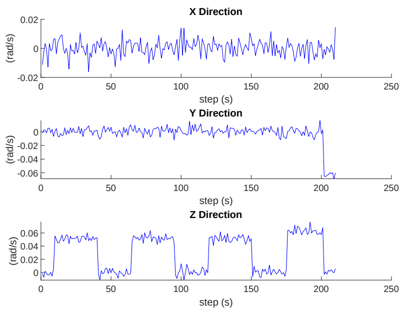
(b) XY plane

Fig. 2: Scenario 1 - Simulation result with the MHE method

During the simulation system disturbances were considered and defined as zero-mean white Gaussian noise with the following standard deviations for the components of the linear velocity, angular velocity and observation readings, respectively, $\sigma_{\xi_v} = 0.015m/s$, $\sigma_{\xi_\omega} = 1deg/s$, and $\sigma_{\xi_y} = 5cm$, which during the simulation period resulted on the values displayed at Figure 3a for the linear velocity, and Figure 3b for the angular velocity.



(a) Linear velocities



(b) Angular velocities

Fig. 3: Input values of linear and angular velocities

Considering a fewer number of landmarks makes the MHE algorithm simpler, and as result more efficient. However, this might result on a worse estimation if the noise regarding the sensors is relatively big. Besides that, the size of the horizon is also a big factor on the computational complexity as it introduces more constraints and estimation variables into the cost function. With that in mind, Table 1 displays the influence of the size of the horizon, H, and number of observed landmarks, n_O, in the average computational time that each step takes using the MATLABs fmincon SQP optimization algorithm, $\bar{t}_{CPU}$ and total vehicle position estimation error, defined as $\tilde{p}_V = \sum_{k=0}^{k_{end}} \mathbf{1}_{1\times3}|\hat{\boldsymbol{p}}_k - \boldsymbol{p}_k|$. Analyzing the Table it is possible to note that the bigger the horizon and the number of landmarks at each step, the better for the MHE to adapt to abrupt dynamics such as the rotation at each edge of the square-shaped corridor.

The optimization algorithm can also have an influence on how fast it reaches a solution. To understand which one would be best suited for this specific application, it was considered an horizon of $H = 10$ and compared the total position estimation error, $\tilde{p}_V$, total landmark position estimation error, $\tilde{p}_{LM} = \sum_{i\in I_L} \sum_{k=0}^{k_{end}} \mathbf{1}_{1\times3}|\hat{\boldsymbol{p}}_{ik} - \boldsymbol{p}_{ik}|$, average computation time and the maximum computation timep for MATLABs' fmincon using active-set, SQP and interior-point algorithms, as demonstrated in Table 2. Other solvers were also included, such as the bnb and bmibnb, which were integrated in MATLAB using the yalmip package [1]. The algorithms Gurobi and Mosek, which have free access for academia,

Table 1: Scenario 1 - Influence of horizon and number o landmarks vs. CPU time and estimated position error (using SQP method).

H	n_O	$\tilde{p}_V$	$\bar{t}_{CPU}$
3	4	0.4789 m	2.086 s
3	8	0.4592 m	4.271 s
5	4	0.4501 m	3.525 s
5	8	0.4445 m	10.803 s
10	4	0.4429 m	22.106 s
10	8	0.4429 m	36.237 s
15	4	0.4429 m	35.004 s
15	8	0.4429 m	47.938 s

Table 2: Scenario 1 - Optimization algorithm influence on position error, sum of landmarks error, and CPU time ($n_O = 4$ and $H = 10$).

Method	$\tilde{p}_V$	$\tilde{p}_{LM}$	$\bar{t}_{CPU}$	$t_{CPU_{max}}$
Active-set	0.4429 m	19.7292 m	30.2316 s	42.2277 s
SQP	0.4429 m	20.1721 m	22.106 s	29.2181 s
interior-point	0.4429 m	19.7292 m	30.527 s	42.3698 s
bnb	0.4429 m	19.923 m	35.342 s	43.4451 s
bmibnb	0.4429 m	19.923 m	38.782 s	45.2945 s

and KKTQP could not deal with the polynomial characteristics of the cost function. The Table 2 shows that the SQP algorithm has not only a lower average processing time per step, but also a lower peak processing time per step, in relation to the other approaches, whilst maintaining an equal or similar error on the estimated final position.

Considering the data displayed at Tables 1 and 2, it is considered that the SQP algorithm with a limit of 4 landmarks at each iteration and an horizon of 3 steps is sufficient to our considered scenario, since the lost in performance to other sets of parameters is not relevant enough.

It was also tested the design of the EKF algorithm in order to guarantee that it was working correctly and as expected. For that, the error and respective 3-sigma bound covariance matrix values at play on the filter were plotted, as displayed in Figure 4. It can be observed that the estimated position error is always within the respective bounds defined, which corroborates that the filter was well designed.

4 Comparative results and discussion

To evaluate the performance of the MHE solution, it was considered the EKF, which is a well-known algorithm for this type of problem, as a solution to the estimation. It was used Scenario 2, as in Figure 5 and it was considered the same

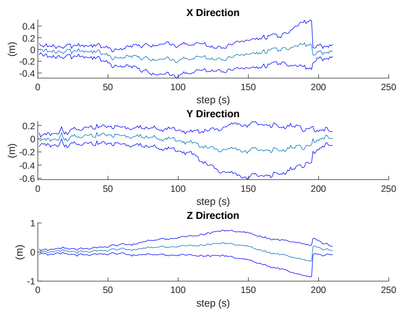


Fig. 4: Scenario 1 - EKF position estimation error (estimate and 3σ bounds).

standard deviations for the disturbances of the linear velocity, angular velocity and observation readings.

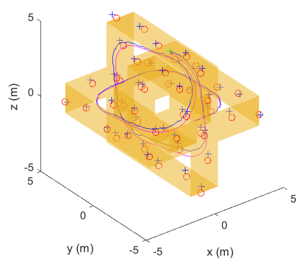
The results of the two algorithms are discussed regarding this new Scenario. The estimated trajectories of both algorithms, MHE and EKF, are depicted in Figures 5 and 6, respectively. On Figure 7 it is displayed the vehicle estimated position and orientation error using the MHE and the EKF algorithms. The orientation error was calculated following

$$e_R = \frac{1}{2} S^{-1}\left[\boldsymbol{R}_q(\boldsymbol{q})^T \boldsymbol{R}_q(\hat{\boldsymbol{q}}) - \boldsymbol{R}_q(\hat{\boldsymbol{q}})^T \boldsymbol{R}_q(\boldsymbol{q})\right] \tag{13}$$

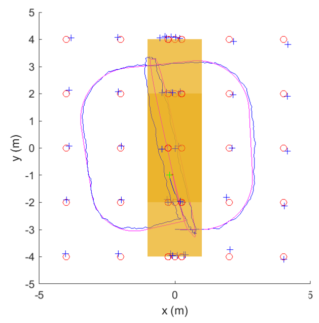
where $\boldsymbol{S}^{-1}[.]$ is the inverse operation of $\boldsymbol{S}[.]$, as $\boldsymbol{S}^{-1}\Big[\boldsymbol{S}[\boldsymbol{\omega}]\Big] \longrightarrow \boldsymbol{\omega}$.

Through these simulations it is possible to analyze that overtime the estimation of the MHE algorithm gets worse than the estimation of the EKF. This can be explained by the capability of the EKF algorithm to do close loops, as it identifies that the landmarks at the end of the first loop are the same as the ones it started the simulation with, which have a smaller covariance associated to them, and corrects the positioning of the vehicle and the previous landmarks. This also happens to an extent in MHE, but it is only able to correct estimations of the previous landmarks that are still on the horizon and gets a worse performance with the increase of the horizon H. Since, the bigger the window is, the less relevance a single landmark will have in the optimal solution for the cost function J, resulting in a tenuous effect on the estimation. However, in the step previous to the close loop event, which happens at step 194, the MHE algorithm has a lower total positional error than the EKF solution. At that moment the sum of the MHE ABS error along the simulation time is 37.77 m, whilst the EKF is 57.45 m. After the complete trajectory these values are 223.85 m and 130.15 m, respectively, as summarized in Table 3. This result shows that the MHE algorithm has a better estimation than the EKF, but it also demonstrates its inability to deal with loop closure (LC) events.

Lastly, the average computation time for each iteration and the total position estimation error before and after the EKF loop closure is presented in Table

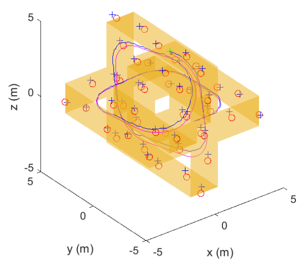


(a) Custom view

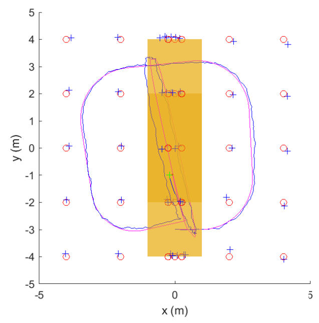


(b) XY plane

Fig. 5: Scenario 2 - MHE-based SLAM method

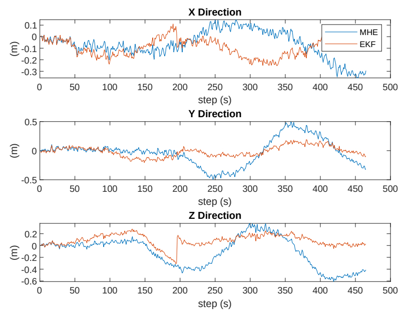


(a) Custom view

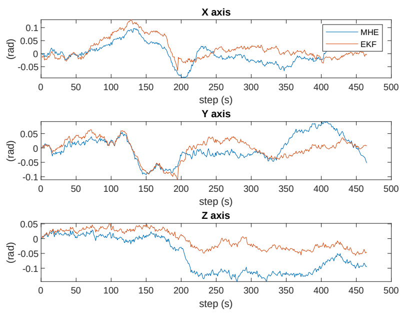


(b) XY plane

Fig. 6: Scenario 2 - EKF-based SLAM method



(a) Vehicle positional error



(b) Orientation error

Fig. 7: Scenario 2 - Comparison of the vehicle positional and orientation error between the MHE and EKF algorithms

3 for both SLAM estimation approaches in Scenario 2. As expected, we can observe that the computation time of the MHE is much bigger when compared to the EKF solution, as the former involves solving an optimization problem for each iteration. Nonetheless, it is expected that using more efficient optimization libraries and development environments, the proposed MHE algorithm can be implemented in real-time.

Table 3: Average CPU time and total position error of EKF and MHE.

Algorithm	$\bar{t}_{CPU}$	$\tilde{p}_V$ (before LC)	$\tilde{p}_V$ (end)
EKF	0.0218 s	37.77 m	223.85 m
MHE	2.974 s	57.45 m	130.15 m

5 Conclusion

This addresses the use of MHE algorithm as a solution to the SLAM problem of vehicles in 3D space with six DoF. Regarding the performance of the MHE it was possible to conclude that the MHE is heavily constrained by the time that it takes to solve its cost function. The best result was achieved with the MATLAB's SQP algorithm, as it not only guaranteed a smaller average processing time, but it also displayed the smallest maximum processing time. Moreover, it was also stated that the size of the considered past data window H, as well as the number of landmarks, also have a big impact on the estimation time, as it introduces more variables and constraints to be estimated, which might result in a stronger non-convex optimization problem. On Scenario 2 it is possible to recognize the importance of the loop closure capability, specially of the EKF, as it is possible to see the correction of its position and landmark estimations. This capability is almost nonexistent on the probabilistic approach to MHE. However, according to the results obtained, it is possible to conclude that the MHE displayed a better estimation when there are no loop closures.

One of the aspects that should be considered in future work is the inability of the probabilistic MHE to correct past estimations when it reaches a loop closure event. A backwards approach to the MHE could be triggered in such events as it is done in other SLAM algorithms. Lastly, a real-time experiment with real data and optimized libraries would be an interesting information to corroborate the results found throughout this work.

Acknowledgments

This work was partially funded by FCT projects CAPTURE (https://doi.org-/10.54499/PTDC/EEI-AUT/1732/2020), CTS (UIDB/00066/2020) and LARSYS (UIDB/50009/2020).

References

1. Yalmip. `https://yalmip.github.io/`. Accessed: 2023-02-12
2. Bemporad, A., Mignone, D., Morari, M.: Moving horizon estimation for hybrid systems and fault detection. In: Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251), vol. 4, pp. 2471–2475. IEEE (1999)
3. Biswas, B., Chatterjee, S., Mukherjee, S., Pal, S.: A discussion on euler method: A review. Electronic Journal of Mathematical Analysis and Applications **1**(2), 294–317 (2013)
4. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: part i. IEEE Robotics & Automation Magazine **13**(2), 99–110 (2006). DOI 10.1109/MRA.2006.1638022
5. Farina, M., Ferrari-Trecate, G., Scattolini, R.: Distributed moving horizon estimation for linear constrained systems. IEEE Transactions on Automatic Control **55**(11), 2462–2475 (2010)
6. Ferrari-Trecate, G., Mignone, D., Morari, M.: Moving horizon estimation for hybrid systems. IEEE transactions on automatic control **47**(10), 1663–1676 (2002)
7. Gallier, J., Gallier, J.: The quaternions and the spaces $\mathbb{S}^3$, su (2), so (3), and $\mathbb{RP}^3$. Geometric Methods and Applications: For Computer Science and Engineering pp. 248–266 (2001)
8. Haseltine, E.L., Rawlings, J.B.: Critical evaluation of extended kalman filtering and moving-horizon estimation. Industrial & engineering chemistry research **44**(8), 2451–2460 (2005)
9. Jørgensen, J.B.: Moving horizon estimation and control. Ph.D. thesis, Technical University of Denmark (2004)
10. Kasahara, T., Tsuno, K., Nonaka, K., Sekiguchi, K.: Comparative experiments of moving horizon estimation based slam in indoor environment. In: 2019 12th Asian Control Conference (ASCC), pp. 1131–1136. IEEE (2019)
11. Kishimoto, Y., Takaba, K., Ohashi, A.: Moving horizon multi-robot slam based on c/gmres method. In: 2019 International Conference on Advanced Mechatronic Systems (ICAMechS), pp. 22–27 (2019). DOI 10.1109/ICAMechS.2019.8861681
12. Muske, K.R., Rawlings, J.B.: Nonlinear moving horizon state estimation. In: Methods of model based process control, pp. 349–365. Springer (1995)
13. Müller, M.A.: Nonlinear moving horizon estimation in the presence of bounded disturbances. Automatica **79**, 306–314 (2017). DOI 10.1016/j.automatica.2017.01.033
14. Rao, C., Rawlings, J., Mayne, D.: Constrained state estimation for nonlinear discrete-time systems: stability and moving horizon approximations. IEEE Transactions on Automatic Control **48**(2), 246–258 (2003). DOI 10.1109/TAC.2002.808470
15. Rao, C.V.: Moving horizon strategies for the constrained monitoring and control of nonlinear discrete-time systems. The University of Wisconsin-Madison (2000)
16. Rao, C.V., Rawlings, J.B.: Constrained process monitoring: Moving-horizon approach. AIChE Journal **48**(1), 97–109 (2002). DOI 10.1002/aic.690480111
17. Rawlings, J.B., Bakshi, B.R.: Particle filtering and moving horizon estimation. Computers & Chemical Engineering **30**(10), 1529–1541 (2006). DOI 10.1016/j.compchemeng.2006.05.031
18. Robertson, D.G., Lee, J.H., Rawlings, J.B.: A moving horizon-based approach for least-squares estimation. AIChE Journal **42**(8), 2209–2224 (1996)