Projectile trajectory estimation: an LSTM approach

Alicia Roux, Jonathan Weber, Jean-Philippe Lauffenburger, Sébastien Changey

Projectile trajectory estimation: an LSTM approach

Alicia ROUX
Guidance, Navigation and Control (GNC) department
French-German Research Institute of Saint-Louis
Saint-Louis, France
alicia.roux@isl.eu

Sébastien CHANGEY
Guidance, Navigation and Control (GNC) department
French-German Research Institute of Saint-Louis
Saint-Louis, France
sebastien.changey@isl.eu

Jonathan WEBER
Université de Haute-Alsace, IRIMAS (UR 7499)
Mulhouse, France
jonathan.weber@uha.fr

Jean-Philippe LAUFFENBURGER
Université de Haute-Alsace, IRIMAS (UR 7499)
Mulhouse, France
jean-philippe.lauffenburger@uha.fr

***Abstract*—This paper presents a deep learning approach to estimate a generic mortar trajectory in a GNSS-denied environment. For this purpose, Long-Short-Term-Memories (LSTMs) are trained on projectile fire simulations. Network input data are the embedded IMU (Inertial Measurement Unit), the reference magnetic field, flight parameters specific to the considered ammunition (initial velocity, fin angle, barrel elevation) and a time vector. This paper focuses on the influence of input data normalization and navigation frame rotation during the training step, leading to rescaling a 3D-value over similar variation ranges with no information loss. LSTM estimates are compared to a classical Dead Reckoning navigation algorithm. Results clearly show the AI contribution, especially for projectile position and velocity estimation.**

***Index Terms*—Projectile navigation, Inertial Measurement Unit, Artificial intelligence, Long-Short-Term-Memory**

I. Introduction

Projectile trajectory estimation is a complex task due to dynamic constraints imposed on the system and low-cost sensors used. For this purpose, projectile navigation is based on embedded IMU (Inertial Measurement Unit) and GNSS (Global Navigation Satellite System) signals. The IMU and GNSS measurements are combined with navigation algorithms such as Kalman Filters [1] to estimate a trajectory. Due to GNSS signals vulnerability (hostile conditions, disturbed or unavailable signals) [2], users aims to exclude these measurements for trajectory estimation [3]–[6].

Moreover, new methods based on AI (Artificial intelligence) are increasingly used for defense applications such as surveillance, reconnaissance, tracking or navigation [7], [8]. Therefore, this paper presents an AI-based algorithm to estimate a projectile trajectory in a GNSS-denied environment using only the embedded IMU and pre-flight parameters specific to the ammunition considered.

Considering that a trajectory is a time serie, AI can provide interesting approaches for its estimation. Time series prediction could be based on Recurrent Neural Networks (RNN) [9]–[11]. RNNs are particularly well suited for time series prediction as they memorize past data to predict future data. However, the simplest form of RNNs, the Vanilla RNN, exhibits vanishing/exploding gradient problems during the training step, so another form of RNNs can be considered: the Long Short-Term Memory (LSTM) [9]–[11]. A LSTM is an extension of the Vanilla RNN including a memory cell in addition to the hidden states, in order to capture both long-term and short-term time dependencies [11].

This paper presents an AI-based solution for projectile navigation in a GNSS-denied environment. LSTMs are trained to estimate projectile position, velocity and Euler angles.
In summary, the main contributions of this work are:

- to detail an LSTM-based approach to estimate a mortar trajectory in the local navigation frame, from IMU measurements, the reference magnetic field, flight parameters (initial velocity, fin angle, barrel elevation) and a time vector.
- to present BALCO (BALlistic COde) [12] used to generate the dataset. This simulator provides true-to-life trajectories of several projectile types according to specific flight parameters. Note that this work focuses only on generic mortar trajectories.
- to investigate different normalization form of the LSTM input data in order to evaluate their contribution on the estimation accuracy. For this purpose, several LSTMs are trained with different input data normalization.
- to study the impact of the local navigation frame rotation on the estimation accuracy. Rotating the local navigation frame during the training step allows a quantity to have similar variation ranges along the three axes.
- to evaluate LSTM estimation accuracy compared to a classical Dead Reckoning navigation algorithm [1], performed on the whole test dataset.

The outline of the paper is as follows. A first part *(II)* presents a brief introduction to projectile navigation and LSTM operating principle. A second part *(III)* focuses on the projectile trajectory dataset. The third part *(IV)* details LSTMs trained to estimate a projectile trajectory and finally, the last part *(V)* presents estimation results.

II. Related work

This section introduces conventional algorithms used for projectile navigation, AI-based navigation approaches, and a brief overview of the LSTM principle.

A. Model-based projectile trajectory estimation

Projectile navigation exploits GNSS (Global navigation satellite system) and IMU (Inertial Measurement Units) measurements, i.e. accelerometers, gyrometers or magnetometers embedded in the projectile. These data are then fused with Kalman Filters [1] such as the Adaptive Extended Kalman Filter [13], the Invariant Extended Kalman Filter [3], the Unscented Kalman Filter [14] or the mixed Extended Unscented Filter [15]. These filters are based on a Dead Reckoning algorithm and then corrected by observations. A Dead Reckoning algorithm [1] aims to integrate gyrometer ω and accelerometer a readings in the sensor frame $\boldsymbol{s}$ to estimate at each discrete time k:

$$R_k = R_{k-1}[\omega_k \Delta_t]_\times \tag{1}$$

$$v_k = v_{k-1} + (R_{k-1}a_k + g)\,\Delta_t \tag{2}$$

$$p_k = p_{k-1} + v_{k-1}\Delta_t + \tfrac{1}{2}(R_{k-1}a_k + g)\,\Delta_t^2 \tag{3}$$

with $R_k \in SO(3)$ the rotation matrix from the sensor frame $\boldsymbol{s}$ to the local navigation frame $\boldsymbol{n}$ determined by the projectile Euler angles, $g \in \mathbb{R}^3$ the constant gravity vector, $p_k \in \mathbb{R}^3$ and $v_k \in \mathbb{R}^3$ respectively the projectile position and velocity, and $[.]_\times$ an $SO(3)$ operator defined as:

$$\forall x \in \mathbb{R}^3, \quad [x]_\times = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{4}$$

Moreover, for high-speed spinning projectiles such as a 155mm shell, the embedded sensors have to resist to high rotation rates and accelerations and some standard sensors such as gyrometers saturate and become useless as explained in [16]. Therefore, several algorithms specialized in high-speed spinning projectile attitude estimation exploit the magnetometer measurements as in [4]–[6].

B. AI-based trajectory estimation

AI methods are increasingly used in the military field such as surveillance and target recognition, military training or cybersecurity [8], [17], [18]. Nevertheless, AI-based projectile trajectory estimation is not commonly used, despite recurrent networks (RNNs) are perfectly adapted to such applications.

Recurrent Neural Networks: Recurrent networks are commonly used for time series prediction as estimations are computed from past characteristics memorized in feedback loops [9]–[11]. The simplest form of RNNs, the Vanilla RNN, has no memory cell and is therefore inadequate to model long-term time dependencies. In addition, this network exhibits vanishing/exploding gradient problems during the training step [9], [11]. To overcome this issue, memory cells have been added to the Vanilla RNN, forming the Long Short-Term Memory (LSTM).

LSTM unit: A LSTM is composed by several units to deal with short and long-term memory. Figure 1 presents a LSTM unit with x_t the input at timestamp t, h_{t-1} the hidden state (previous LSTM unit output) and c_t the memory state, which aims to memorize long-term dependencies.

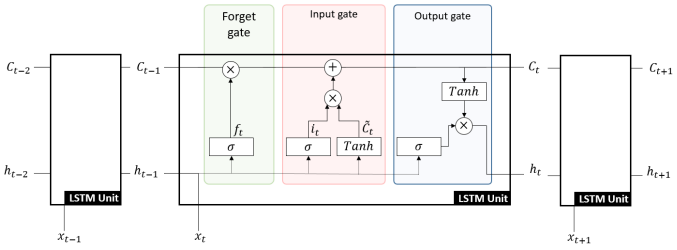

Fig. 1. LSTM cell operating principle composed by three gates with x_t the input at the current time, h_{t-1} the hidden state at the previous time, and c_t the memory cell state.

As shown in Figure 1, a LSTM unit is composed by three gates:
– the *forget gate* filters, through a Sigmoid function σ, data contained in the concatenation of x_t and h_{t-1}. Data are forgotten for values close to 0 and are memorized for values close to 1. The *forget gate* model is:

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{5}$$

– the *input gate* extracts relevant information from $[h_{t-1}, x_t]$ by applying a Sigmoid σ and a Tanh function. The *input gate* is represented by:

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_f) \tag{6}$$

$$\tilde{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{7}$$

The memory cell C_t is updated from the *forget gate* f_t and the *input gate* i_t, $\tilde{C}_t$, to memorize pertinent data:

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{8}$$

– the *output gate* defines the next hidden state h_t containing information about previous inputs. The hidden state h_t is updated with the memory cell C_t normalized by a Tanh function and $[h_{t-1}, x_t]$ normalized by a Sigmoid function:

$$h_t = \sigma(W_h.[h_{t-1}, x_t] + b_h) \times tanh(C_t) \tag{9}$$

with $W_{(.)}$ and $b_{(.)}$, the different gate weights and biases.

Currently, only a few works exploit recurrent networks in the military field. There are commonly used for navigation, such as aircraft flight path prediction [19], [20], vehicle trajectory estimation [21], maritime route prediction [22], human motion prediction [23], [24] or target tracking [25]. It is however interesting to mention [26] focusing on projectile trajectory estimation based on LSTMs trained from incomplete and noisy radar measurements.

III. Projectile trajectory dataset

Results presented in this paper exploit a projectile fire dataset generated by BALCO (BALlistic COde) [12], i.e. a high fidelity projectile trajectory simulator.

As shown in Figure 2, two reference frames are considered:
– the *local navigation frame* ***n*** (black frame in Figure 2) in which projectile trajectories are expressed, is tangent to the Earth and assumed fixed during the projectile flight.
– the *sensor frame* ***s*** (green frame in Figure 2), rigidly fixed to the projectile and misaligned with the projectile gravity center, the frame where the inertial measurements are performed.

The dataset used in this work includes 5 000 mortar fire simulations and each simulation includes:

- *IMU measurements in the sensor frame* ***s****:* gyrometer $\omega \in \mathbb{R}^3$, accelerometer $a \in \mathbb{R}^3$ and magnetometer $h \in \mathbb{R}^3$ readings. Two kinds of inertial measurements are available:
  – *IMU measurements* performed in the sensor frame including a sensor error model: a misalignment model between each sensor axis and the projectile gravity center, a sensitivity factor, a bias and a noise (assumed Gaussian with zero mean) relative to each sensor axis.
  – *IMU DYN measurements*, performed in the sensor frame and issued from *IMU measurements* to which a transfer function is added to each sensor. This sensor model allows to denotes the response of the three sensors over the operating range.
- *reference magnetic field* $h_n \in \mathbb{R}^3$*:* in the local navigation frame ***n***, assumed constant during the projectile flight.
- *flight parameters:* the fin angle $\delta_f \in [0, 3]°$, the initial velocity at the end of the propulsion phase $v_0 \in [220, 320]$ m/s, and the barrel elevation angle $\alpha \in [40, 60]°$.
- *time vector* $k\Delta_t$ with $\Delta_t = 1e^{-3}s$ the IMU sampling period.
- *reference trajectory:* projectile position p, velocity v and Euler angles Ψ in the local navigation frame ***n*** at the IMU frequency.

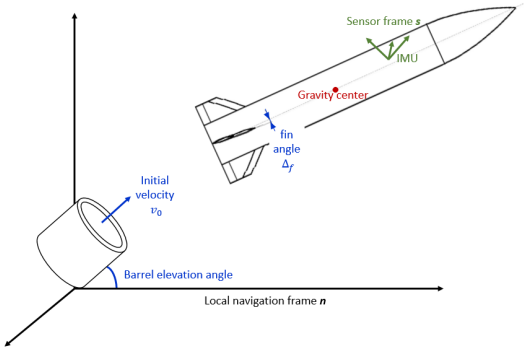

Fig. 2. Navigation frames (black - Local navigation frame ***n***, red - projectile gravity center, green - sensor frame ***s***) and flight parameters (fin angle δ_f, initial velocity v_0, barrel elevation angle α).

IV. Problem formulation

This part introduces LSTMs trained to estimate projectile trajectories from the dataset presented in section *(III)*. Four network types derived in eight versions are trained. Moreover, this part presents normalization methods, the local navigation frame rotation and network training characteristics.

A. Overview

As mentioned in the introduction *(I)*, the LSTM goal is to estimate a projectile trajectory from pre-flight data and the embedded IMU. As shown in Figure 3, LSTM predictions at time t are obtained from an input sequence of length τ and where the 16 features are:

- inertial measurements $\mathcal{M} \in \mathbb{R}^{12}$: *IMU measurements*, i.e. accelerometer a, gyrometer ω and magnetometer h readings in the sensor frame ***s*** and the reference magnetic field h_n in the local navigation frame ***n***,
- flight parameters $\mathcal{P} = (\delta_f, v_0, \alpha) \in \mathbb{R}^3$,
- the time vector $\mathcal{T} \in \mathbb{R}^1$.

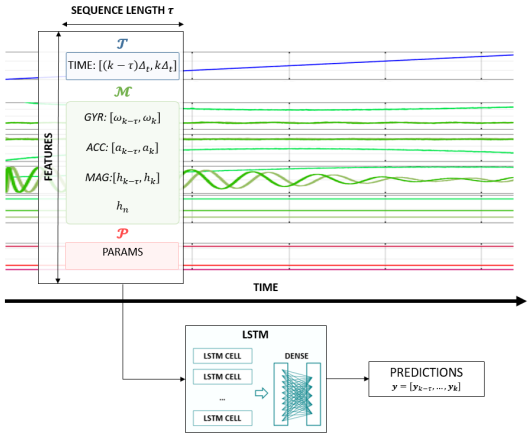

Fig. 3. LSTM input data: inertial measurements $\mathcal{M}$, flight parameters $\mathcal{P}$, time vector $\mathcal{T}$.

Four LSTMs types are trained and differ depending on the output features learned. $LSTM_{ALL}$ trained to estimate 9 output features which are the projectile position p, velocity v and Euler angles Ψ in the local navigation frame ***n***, $LSTM_{POS}$, $LSTM_{VEL}$, $LSTM_{ANG}$ trained to estimate 3 output features which are respectively the projectile position p, the projectile velocity v and the projectile Euler angles Ψ in the local navigation frame ***n***.

LSTMs are declined in 8 versions presented in Table I, to study the influence of the Min/Max $MM(.)$ and the Standard Deviation $STD(.)$ normalization as well as the influence of the local navigation frame rotation on estimation accuracy.

B. Input data normalization

Network input data normalization is a preprocessing data approach to rescale input data to similar variation ranges while preserving the same distribution and ratios as the original data.

TABLE I
VERSION SPECIFICATIONS: INFLUENCE OF THE NETWORK INPUT DATA NORMALIZATION AND THE LOCAL NAVIGATION FRAME ROTATION.

Name	NORMALIZATION	ROTATION
V_1	No	No
V_2	$MM(\mathcal{T}), MM(\mathcal{M}), MM(\mathcal{P})$	No
V_3	$MM(\mathcal{T}, \mathcal{M}, \mathcal{P})$	No
V_4	$STD(\mathcal{T}), STD(\mathcal{M}), STD(\mathcal{P})$	No
V_5	$STD(\mathcal{T}, \mathcal{M}, \mathcal{P})$	No
V_6	No	Yes
V_7	$MM(\mathcal{T}, \mathcal{M}, \mathcal{P})$	Yes
V_8	$STD(\mathcal{T}, \mathcal{M}, \mathcal{P})$	Yes

Min/Max normalization: Versions V_2, V_3 and V_7 use the Min/Max normalization $MM(.)$ defined as follows:

$$x_{MM} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{10}$$

with x_{max} and x_{min} respectively the maximum and minimum of x. This normalization ranges values in the interval $[0, 1]$.

Standard Deviation normalization: Versions V_4, V_5 and V_8 use the Standard Deviation normalization $STD(.)$ defined as follows:

$$x_{STD} = \frac{x - \mu}{\sigma} \tag{11}$$

with x the quantity to normalize, μ its mean and σ its standard deviation. Thus x_{STD} is a quantity with a zero-mean and a standard deviation of one. This normalization is especially used for input data with different units.

It is important to note that the normalization factors x_{max}, x_{min}, x_{min}, μ and σ are computed before networks training, and are evaluated on the training dataset.

C. Local navigation frame rotation

As presented in Table I, versions V_6, V_7 and V_8 use the local navigation frame rotation. This method, illustrated in Figure 4, aims to rotate the local navigation frame $\boldsymbol{n}$ by a fixed angle γ (local rotated navigation frame $\boldsymbol{n_\gamma}$) such as:

$$x_\gamma = R_\gamma x \tag{12}$$

with $x \in \mathbb{R}^3$ defined in $\boldsymbol{n}$, $x_\gamma \in \mathbb{R}^3$ expressed in $\boldsymbol{n_\gamma}$ and $R_\gamma \in SO(3)$ the transition matrix from the local navigation frame $\boldsymbol{n}$ to the local rotated navigation frame $\boldsymbol{n_\gamma}$. The rotation is applied along the three local navigation frame axes and the angle γ is fixed manually to ensure that the three components of a quantity expressed in the local rotated navigation frame $\boldsymbol{n_\gamma}$ are similar along the three axes. In other words, the local navigation frame rotation is used for the same purpose as normalization, to express a quantity over similar variation ranges.

For example, projectile position variation ranges along the x-axis and z-axis are around several kilometers while along the y-axis, the projectile position varies by a few meters. Express the position in the local rotated navigation frame $\boldsymbol{n_\gamma}$, the position along the three axes are around one kilometer.

All quantities expressed in the local navigation frame $\boldsymbol{n}$ are rotated, i.e. projectile position, velocity and Euler angles.

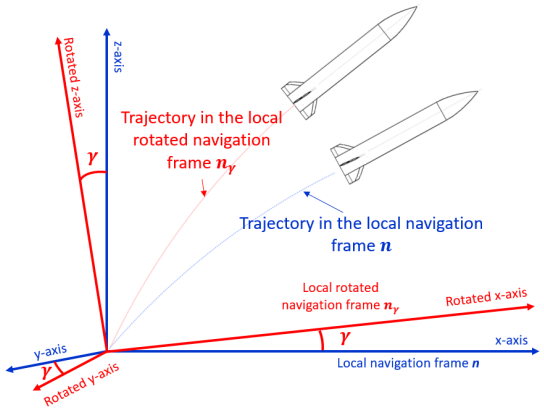

Fig. 4. Local navigation frame **n** and local rotated navigation frame $\boldsymbol{n_\gamma}$.

In other words, during the training step, labels are expressed in the local rotated navigation frame $\boldsymbol{n_\gamma}$ and LSTMs predict trajectories in $\boldsymbol{n_\gamma}$. During testing, LSTMs estimate projectile trajectories in the local rotated navigation frame $\boldsymbol{n_\gamma}$ and then, estimations are moved back to the initial local navigation frame $\boldsymbol{n}$.

D. LSTM training details

Networks $LSTM_{ALL,\ POS,\ VEL,\ ANG,\ V_{1-8}}$ are trained on a training dataset composed by 100 simulations, a validation dataset composed by 10 simulations and a test dataset composed by 20 simulations generated by BALCO presented in part *(III)*. This reduced dataset is defined to evaluate the impact of the normalization and the local navigation frame rotation on estimation accuracy. Moreover, one simulation includes an average of 35 000 time steps.

The batch size is 64 and the window size (SEQ_LEN) is set to 20 timestamp to capture enough long-term dependencies without depending on measurement noise. LSTMs are composed of two layers of 64 and 128 hidden units.

The loss between LSTM estimates and the reference trajectory is evaluated with the Mean Squared Error (MSE) defined as:

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(\widehat{x}_k - x_{ref_k})^2 \tag{13}$$

with $\widehat{x}$ the LSTM estimate and x_{ref} the reference value. Network weights and biases are updated by Adam optimization algorithm [27].

V. RESULTS AND ANALYSIS

This section reports the estimated trajectories of a mortar according to the different networks mentioned in section *(IV)*. LSTM estimates are compared to a classical navigation algorithm: a Dead Reckoning (1)-(3).
A first part *(V-A)* focuses on one mortar trajectory result, then, the LSTM performances are analyzed on the whole test dataset *(V-B)*. Finally, the last part *(V-C)* is centered on the LSTM performance on a larger dataset and on the influence of the IMU error model.

A. Focus on one mortar fire simulation

Figures 5-7 present the estimated position, velocity, and Euler angles and the associated errors for one mortar shot in the test dataset. For readability reasons, three estimation methods are first compared: the Dead Reckoning algorithm (1)-(3), $LSTM_{ALL,V_1}$ *(IV)*, and $LSTM_{ALL,V_6}$ (local navigation frame rotation).

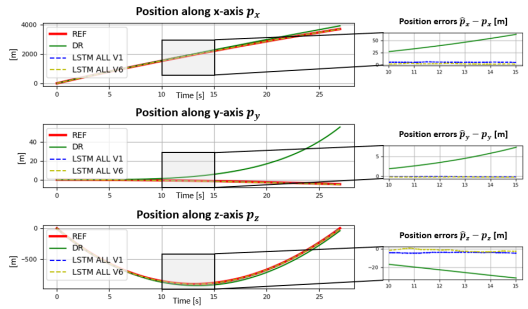

Fig. 5. Estimated projectile position and associated errors [m].

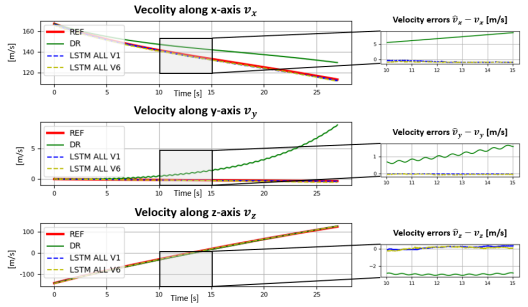

Fig. 6. Estimated projectile velocity and associated errors [m/s].

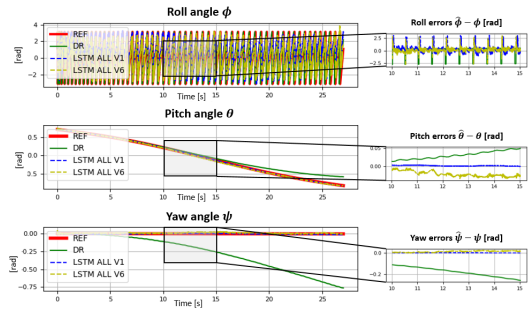

Fig. 7. Estimated projectile Euler angles and associated errors [rad].

As shown in Figures 5 and 6, positions and velocities estimated by LSTMs are significantly more accurate than Dead Reckoning. For the orientation (Figure 7), LSTMs are only precise to estimate the pitch θ and yaw angle ψ. Errors in LSTM roll angle ϕ estimation are due to mortar rotation rate. LSTMs fail to fully capture all roll angle variations. Moreover, according to Figures 5 and 6, rotating the local navigation frame improves projectile position and velocity estimation but slightly degrades pitch angle θ estimation (Figure 7).

B. Analysis on the whole test dataset: 20 mortar shots

In order to validate the previous observations, networks presented in section *(IV)*, $LSTM_{ALL,\ POS,\ VEL,\ ANG,\ V_{1-8}}$, are evaluated on the whole test dataset according to two criteria based on the Root Mean Square Error (RMSE):

$$RMSE_x = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(\widehat{x}_k - x_{k,ref}\right)^2} \tag{14}$$

with $\widehat{x}$ the estimated quantity, x_{ref} the reference and N the number of samples.

Success Rate $\mathcal{C}_1$: The first evaluation criterion is the success rate, i. e. number of simulations in the test dataset where a LSTM's RMSE is strictly smaller than the Dead Reckoning's RMSE:

$$\mathcal{C}_1 = \sum_{k=1}^{N_{sim}} RMSE_{LSTM} < RMSE_{DR} \tag{15}$$

with N_{sim} the number of simulations in the test dataset.

Error Rate $\mathcal{C}_2$: The second evaluation criterion is the error rate evaluated such as:

$$\mathcal{C}_2 = \frac{1}{N_{sim}}\sum_{k=1}^{N_{sim}} RMSE_{sim_k} \tag{16}$$

with N_{sim} the number of simulations in the test dataset.

The success rate $\mathcal{C}_1$ of $LSTM_{ALL,POS,VEL,ANG,V_1-V_8}$ are presented in Figure 8-10.(a) and the error rates of LSTMs $\mathcal{C}_{2_{LSTM}}$ and Dead Reckoning $\mathcal{C}_{2_{DR}}$, are presented in Figure 8-10.(b), for position, velocity and Euler angles estimation.

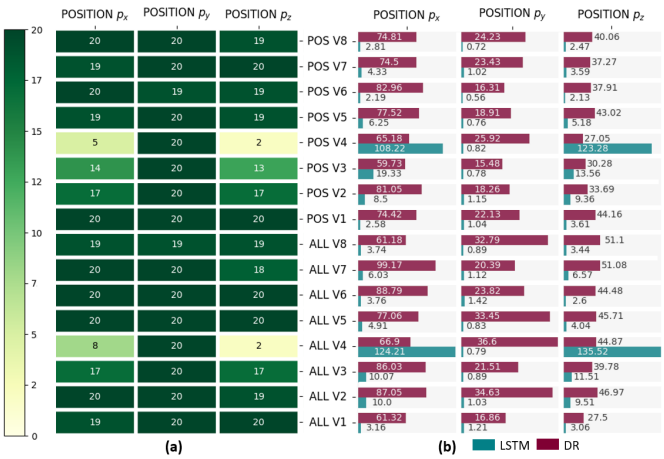

Fig. 8. Position estimation analysis: $\mathcal{C}_1$ criterion *(a)* and $\mathcal{C}_2$ criterion $[m]$ *(b)*.

Position analysis results: According to Figure 8, LSTMs strongly outperform Dead Reckoning for position estimation. Moreover, $LSTM_{POS,V_1-V_8}$, specialized in position estimation exclusively, slightly exceed $LSTM_{ALL,V_1-V_8}$. Normalizations applied to input data affect position estimates differently. Firstly, V_3 and V_4 versions exhibit lower success and error rates than other normalizations. Thus, normalizations

$MM(\mathcal{T},\mathcal{M},\mathcal{P})$ and $STD(\mathcal{T}), STD(\mathcal{M}), STD(\mathcal{P})$ are not appropriate to this application. Secondly, the accuracy of networks with normalization (Min/Max $V_{2,3,7}$ and STD $V_{4,5,8}$) is worse than networks with no normalization $V_{1,6}$ as normalization implies a loss of information. Finally, the Min/Max normalization per feature V_2 is better than a Min/Max normalization for all features V_3, in contrast to the STD normalization (V_5 better than V_4).
Rotating the local navigation frame V_{6-8} improves the position estimation accuracy especially along the z-axis.

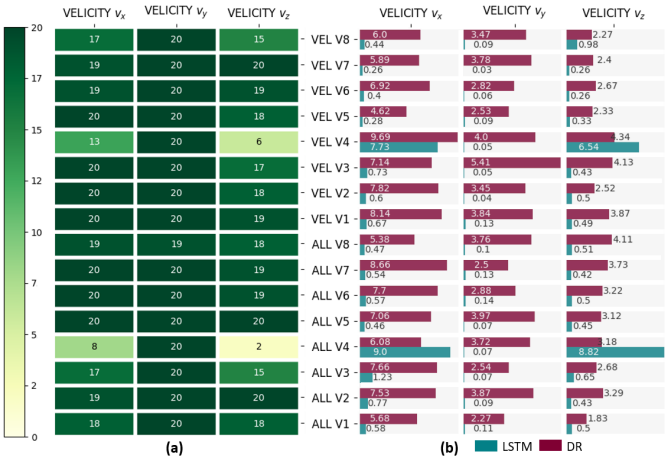

Fig. 9. Velocity estimation analysis: $\mathcal{C}_1$ criterion *(a)* and $\mathcal{C}_2$ criterion $[m/s]$ *(b)*.

Veclocity analysis results: According to Figure 9, similar observations as previously can be formulated. LSTMs clearly outperform Dead Reckoning for velocity estimation. Specialized networks $LSTM_{VEL}$ are a bit better than $LSTM_{ALL}$. The STD normalization for all features V_5 exhibits the best results among the different normalization options investigated, especially for velocity along the z-axis. Moreover, rotating the local navigation frame V_6 significantly improves the projectile velocity estimation along all the three axes.

Fig. 10. Euler angles estimation analysis: $\mathcal{C}_1$ criterion *(a)* and $\mathcal{C}_2$ criterion $[rad]$ *(b)*.

Euler angles analysis results: Euler angles estimation is more mitigated according to figure 10. Focusing on the success rate $\mathcal{C}_1$, LSTMs deteriorate the roll ϕ and pitch θ angles estimates compared to Dead Reckoning, but accurately estimate the yaw angle ψ. In addition, $LSTM_{ANG}$ is less accurate than $LSTM_{ALL}$ for roll ϕ and pitch θ estimation according to criterion $\mathcal{C}_1$, contrary to the yaw angle ψ. As previously, the $STD(\mathcal{T},\mathcal{M},\mathcal{P})$ normalization of $LSTM_{ALL}$ exhibits the best performances for the three Euler angles estimation as well as the local navigation frame rotation.

In summary, end-to-end estimation using LSTM is particularly appropriate for projectile position and velocity estimation. According to the reported results, Figure 8-10, specialized networks do not significantly improve estimation accuracy. Moreover, these results show that $STD(\mathcal{T},\mathcal{M},\mathcal{P})$ normalization is more appropriate to estimate a projectile trajectory. Finally, rotating the local navigation frame is an efficient method to optimize projectile position and velocity estimation.

C. IMU measurements: effects on position estimation

The dataset presented in section *(III)* contains two kinds of inertial readings; *IMU measurements*, used so far, and *IMU DYN measurements*, where sensors are characterized by a dynamic model. This part focuses on the effect of the IMU error model on projectile position estimation. To this end, two LSTMs are trained with the same specifications as $LSTM_{ALL\ V1}$ (no normalization, no rotation):

- $LSTM_{IMU}$ trained with *IMU measurements* with no normalization and no rotation in order to estimate the projectile position, velocity and orientation.
- $LSTM_{IMU\ DYN}$ trained with *IMU DYN measurements* with the same characteristics as $LSTM_{IMU}$.

Both networks are trained on 4 000 mortar fire simulations, validated on 400 and tested on 400.

– $\boldsymbol{LSTM_{IMU}}$ **:** Figure 11 presents the RMSE (14) returned by $LSTM_{IMU}$ as a function of the Dead Reckoning RMSE for projectile position estimation on the 400 simulations in the test dataset.

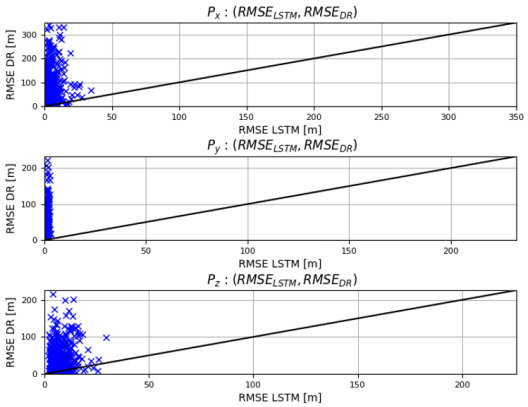

Fig. 11. $LSTM_{IMU}$: $(RMSE_{LSTM}, RMSE_{DR})$ [m] for 400 mortar fire simulations.

According to Figure 11, $LSTM_{IMU}$ significantly outperforms Dead Reckoning for position estimation along the three axes, as most of the markers are located in the upper part. $LSTM_{IMU}$ accurately estimates the projectile position, especially along the y-axis.

Figure 12 presents position errors along at the impact point (position errors (p_x, p_y) at the final time of a shot) for all 400 simulations in the test dataset.

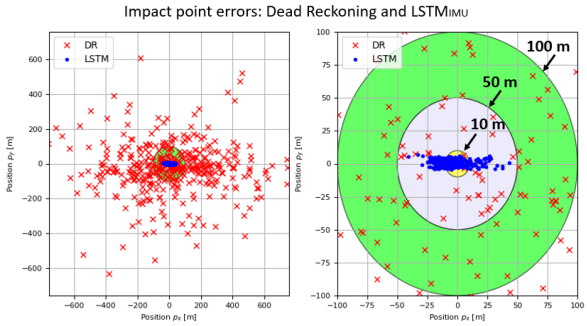

Fig. 12. Errors at impact point obtained by $LSTM_{IMU}$ (blue dot) and the Dead Reckoning (red dot).

As shown in figure 12, most $LSTM_{IMU}$ errors at the impact point are located inside a 50-meter error disk (gray disk), while Dead Reckoning errors are much larger. More precisely, 249 of the 400 shots have errors at the impact point less than 10m with the LSTM, 149 shots with errors between 10 and 50 m and 2 shots with errors between 50 and 100 m. Concerning the Dead Reckoning, 293 of the 400 simulations have errors greater than 100 m. Thus, on a large dataset, with various flight parameters, $LSTM_{IMU}$ succeeds in accurately estimate the projectile position.

– $\boldsymbol{LSTM_{IMU\ DYN}}$ **:** Figures 13 and 14 respectively show $(RMSE_{LSTM_{IMU\ DYN}}, RMSE_{DR})$ for projectile position estimation and errors at the impact point.

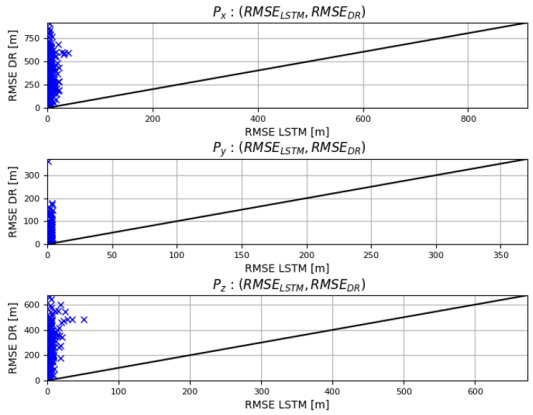

Fig. 13. $LSTM_{IMU\ DYN}$: $(RMSE_{LSTM}, RMSE_{DR})$ [m] for 400 mortar fire simulations.

Dead Reckoning completely diverges for position estimation from IMU DYN data. Conversely, LSTM accurately estimates

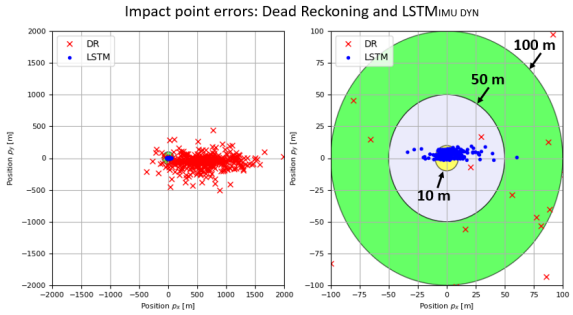

Fig. 14. Errors at impact point obtained by $LSTM_{IMU\ DYN}$ (blue dot) and the Dead Reckoning (red dot).

the projectile position: from Figure 13, most of the markers are located in the upper part, and from Figure 14, most errors at impact point are less than 50m. Concerning the $LSTM_{IMU\ DYN}$, 266 of the 400 shots have errors at the impact point less than 5m while with the Dead Reckoning, 378 of the 400 shots have errors greater than 100m. Therefore, despite sensor dynamics, the LSTM is still able to estimate the projectile position.

In summary, according to Figures 11 and 14, a LSTM is able to accurately estimate a projectile position despite dynamic inertial data and a various range of flight characteristics. In addition, generic guided mortar accuracy is around 10 to 70m and LSTM estimation results improve these accuracies.

Conclusion

This paper presents a deep learning approach to estimate a mortar trajectory in a GNSS-denied environment. LSTMs are trained only from inertial measurements, flight parameters and a time vector. Different normalizations are applied to input data as well as the local navigation frame rotation during the training step, in order to deal with the different variation ranges along the three axes.

According to the reported results, LSTMs are accurate to estimate projectile positions and velocities compared to a classical navigation algorithm. Moreover, LSTMs are still accurate to estimate projectile positions even with dynamic inertial measurements and outperform the accuracy of generic guided mortars.

Currently, this estimation method aims to be tested on real data. Moreover, although not presented here, this method is generalized to other kinds of projectiles: 155mm shells, 40mm projectile and Basic Finner.

Now, the idea is to develop Deep Kalman filters by integrating LSTM models into a Kalman filter. In other words, the idea is to use a LSTM to estimate one model of the Kalman filter such as the prediction model or the observation model. For example, in the case of an Extended Kalman Filter to estimate a projectile trajectory from IMU measurements and corrected by GNSS observations, LSTM estimates can be used as GNSS measurements, allowing to bypass the GNSS and avoid any decoying or jamming problem.

REFERENCES

[1] Paul Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems, Second Edition*. 2013.

[2] Gregory Duckworth and Edward Baranoski. Navigation in GNSS-denied environments: Signals of opportunity and beacons. In *Proceedings of the NATO Research and Technology Organization (RTO) Sensors and Technology Panel (SET) Symposium*, 2007.

[3] Alicia Roux, Sébastien Changey, Jonathan Weber, and Jean-Philippe Lauffenburger. Projectile trajectory estimation: performance analysis of an Extended Kalman Filter and an Imperfect Invariant Extended Kalman Filter. In *2021 9th International Conference on Systems and Control (ICSC)*, pages 274–281, 2021.

[4] Christophe Combettes, Sébastien Changey, Ronan Adam, and Emmanuel Pecheur. Attitude and velocity estimation of a projectile using low cost magnetometers and accelerometers. In *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 650–657, 2018.

[5] Sébastien Changey, Emmanuel Pecheur, Loic Bernard, and all. Real time estimation of projectile roll angle using magnetometers: In-flight experimental validation. In *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, pages 371–376, 2012.

[6] Aurélien Fiot, Sébastien Changey, and Nicolas Petit. Attitude estimation for artillery shells using magnetometers and frequency detection of accelerometers. *Control Engineering Practice*, 122:105080, 2022.

[7] Adrian Carrio, Carlos Sampedro, Alejandro Rodriguez-Ramos, and Pascual Campoy. A review of deep learning methods and applications for unmanned aerial vehicles. *Journal of Sensors*, 2017, 2017.

[8] Peter Svenmarck, Linus Luotsinen, Mattias Nilsson, and Johan Schubert. Possibilities and challenges for artificial intelligence in military applications. 05 2018.

[9] Alex Sherstinsky. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[10] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: LSTM cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[11] Ralf Staudemeyer and Eric Rothstein Morris. Understanding LSTM–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*, 2019.

[12] Pierre Wey, Daniel Corriveau, Thomas Saitz, Wim de Ruijter, and Peter Strömbäck. BALCO 6/7-DoF Trajectory Model. 05 2016.

[13] Hui Zhao and Zhong Su. Real-time estimation of roll angle for trajectory correction projectile using radial magnetometers. *IET Radar, Sonar & Navigation*, 14(10):1559–1570, 2020.

[14] Liangliang An, Liangming Wang, Ning Liu, Jian Fu, and Yang Zhong. A Novel Method for Estimating Pitch and Yaw of Rotating Projectiles Based on Dynamic Constraints. *Sensors*, 19(23), 2019.

[15] Sébastien Changey and all. A mixed extended-unscented filter for attitude estimation with magnetometer sensor. In *2006 American Control Conference*, pages 6 pp.–, 2006.

[16] Nabil Jardak, Ronan Adam, and Sebastien Changey. A Gyroless Algorithm with Multi-Hypothesis Initialization for Projectile Navigation. *Sensors*, 21:7487, 11 2021.

[17] YuLong Zhang, ZiJie Dai, LongFei Zhang, ZhengYi Wang, Li Chen, and YuZhen Zhou. Application of artificial intelligence in military: From projects view. In *2020 6th International Conference on Big Data and Information Analytics (BigDIA)*, pages 113–116, 2020.

[18] Antonio Carlo. Artificial intelligence in the defence sector. In *International Conference on Modelling and Simulation for Autonomous Systems*, pages 269–278. Springer, 2020.

[19] Zhiyuan Shi, Min Xu, Quan Pan, Bing Yan, and Haimin Zhang. LSTM-based Flight Trajectory Prediction. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018.

[20] Nikolai Gaiduchenko, Pavel Gritsyk, and Yanka Malashko. Multi-step ballistic vehicle trajectory forecasting using deep learning models. In *2020 International Conference Engineering and Telecommunication (En&T)*, pages 1–6, 2020.

[21] SeongHyeon Park, Byeongdo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. *CoRR*, abs/1802.06338, 2018.

[22] Kristian Aalling Sørensen, Peder Heiselberg, and Henning Heiselberg. Probabilistic maritime trajectory prediction in complex scenarios using deep learning. *Sensors*, 22(5), 2022.

[23] Abdulrahman Al-Molegi, Mohammed Jabreel, and Baraq Ghaleb. Stf-rnn: Space-time features-based recurrent neural network for predicting people's next location. 12 2016.

[24] Emad Barsoum, John Kender, and Zicheng Liu. Hp-gan: Probabilistic 3d human motion prediction via gan. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[25] Sami Jouaber, Silvere Bonnabel, Santiago Velasco-Forero, and Marion Pilte. NNAKF: A neural network adapted Kalman filter for target tracking. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4075–4079, Toronto, France, June 2021. IEEE.

[26] Li-he Hou and Hua-jun Liu. *An End-to-End LSTM-MDN Network for Projectile Trajectory Prediction*, pages 114–125. 11 2019.

[27] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.