Measurement 182 (2021) 109757

Contents lists available at ScienceDirect

Measurement

journal homepage: www.elsevier.com/locate/measurement

An FPGA-based architecture for a latitude and longitude correction in autonomous navigation tasks

Pedro J. Correa-Caicedo [a], Alejandro I. Barranco-Gutiérrez [a,b], Erick I. Guerra-Hernandez [c], Patricia Batres-Mendoza [c], Jose A. Padilla-Medina [a], Horacio Rostro-González [d,*]

[a] Tecnológico Nacional de México en Celaya, Celaya 38010, Mexico
[b] Catedrás CONACYT
[c] Escuela de Sistemas Biológicos e innovación Tecnológica, Universidad Autónoma Benito Juarez de Oaxaca, Oaxaca 68120, Mexico
[d] Departamento de Electrónica, Universidad de Guanajuato, Salamanca 36885, Mexico

ARTICLE INFO

Keywords:
FPGA-based Implementation
Fuzzy Systems
GPS
Data Correction

ABSTRACT

The response speed of the intelligent systems embedded in an autonomous vehicle is crucial for its correct operation and reduction of the risks on the road derived from autonomous driving. For that reason, it is necessary to optimize the algorithms that process the data from the sensors; with that aim the Field-Programmable Gate Arrays (FPGAs) offer the possibility of parallelizing the tasks to be carried out by mentioned systems, accelerating their response and improving their performance. In this regard, this paper introduces a fuzzy absolute position correction system, which corrects the latitude and longitude data registered from a GPS Pmod sensor and its implementation on a FPGA to speed up the correction results. A necessary comparison of the algorithm execution time on different platforms such as: A Raspberry pi 4 model B, a personal computer (PC) with Ubuntu 18.04.4 64-bit and the FPGA model, was performed to validate the results and the effectiveness of the implementation. The correction system was validated in software and hardware on 4 different routes, each of them with a large number of samples. The results were highly similar in the three platforms; however, the FPGA-based implementation offers a speed up of 40000x compared to software-based implementations.

1. Introduction

One of the main modules of autonomous land vehicle navigation systems is absolute location, which in most applications consists of fusion of data from GPS sensors, odometers and inertial measurement units [1–2]. Despite the great advances in the development of these technologies, the data supplied by these devices maintain a considerable noise level and probability of failure, considering the importance of locating the vehicle accurately and consistently. To solve this problem or at least mitigate the inherent error of these sensors, different absolute position correction algorithms have been developed, such as the Kalman filter and its variations [3].

The uses of neural networks that automatically modify the Kalman filter have been applied to improve its correction in an adaptive way [4]. This article proposes the use of fuzzy logic to create a system capable of mitigating the error of latitude and longitude measurements delivered by the Pmod GPS sensor and at the same time emphasizing the importance of the processing speed in the calculations. The latest because data correction must to be done in real-time for autonomous vehicles to know the current state of the car always, and therefore the need to consider the implementation of the fuzzy correction system in an FPGA.

FPGAs became one of the first options to optimize intelligent systems thanks to the possibility of reprogramming their hardware as appropriate and the parallel processing of data and operations. In [5], the authors propose the implementation of an autonomous navigation system for a vehicle to be used in a robot that inspects structural defects of buildings; highlights the benefits of FPGAs in obtaining fast system response.

In [6], authors propose the design of a reconfigurable mobile robot which is located and tracked using an algorithm implemented in FPGA that processes data from sensors, such as gyroscopes and cameras, achieving an almost completely autonomous navigation system. Besides, in [7] a prototype GPS receiver is proposed using a Xilinx System Generator (XSG) and implemented in FPGA, the reception of the GPS signal is also critical to obtain a reliable absolute location of an autonomous vehicle which highly depends on the number of satellites that the

* Corresponding author.
E-mail address: hrostrog@ugto.mx (H. Rostro-González).

https://doi.org/10.1016/j.measurement.2021.109757
Received 25 January 2021; Received in revised form 1 June 2021; Accepted 11 June 2021
Available online 15 June 2021

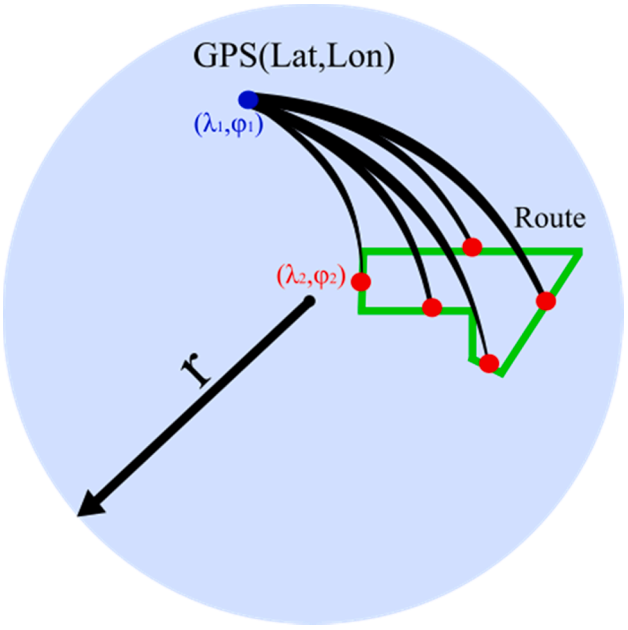

Fig. 1. Distance from a GPS point to the straight lines of a route (Haversines).

receiver manages to capture in a certain geographical area, then FPGAs offer a great advantage by processing this data in parallel and delivering information more quickly. In [8] an object detection and movement tracking system of a mobile robot implemented in FPGA is also shown, obtaining satisfactory results in the differentiation of the objects colors objects and a more precise relative position of the robot. Finally, in [9] an intelligent algorithm is proposed based on the calculation of the Manhattan distance between the objective histogram and the images of pedestrians taken in real time, all this implemented in parallel processing in an FPGA achieving a precision of 99.2% in pedestrians' detection.

Another FPGA implementation of systems focused on autonomous vehicles is presented in [10], where the authors developed a visual perception algorithm for object detection on the road; likewise, in [11], a hardware implementation of a communication system between autonomous vehicles that takes advantage of the benefits of an FPGA to handle large amounts of data is shown. In [12–13], pulse synchronization systems for GPS sensors implemented in FPGA are designed, offering faster update rates with respect to microcontroller implementations.

Currently, there are fuzzy systems for autonomous vehicle applications implemented in FPGA; such as the one presented in [14], where the authors develop a driving assistance system based on a neuro-fuzzy intelligent sensor that provides a high-speed response in real time. In [15], a fuzzy data fusion system for the control of a mobile robot called FBAFC is developed and implemented in an FPGA. In [16], the authors propose the fuzzy digital phase-locked loop (DPLL) system that improves the reception performance of signals coming from GPS satellite networks by modifying parameters such as phase and frequency step signals.

On the other hand, there are other types of developments, such as those introduced in [17–21], which are not directly applied to autonomous vehicles but are related to it. In these papers, authors propose control strategies using fuzzy logic and their respective implementation on an FPGAs. These strategies have motivated the control system presented in this work.

One of the highlights of the present work is that the fuzzy systems developed in this research show to be less susceptible to parameter tuning and simpler operation using only GPS sensor data (in the task of localization and tracking of ground vehicles) being an alternative to the Kalman filter and its variations. It is important to mention this fact without leaving aside the main objective of this article, which is the implementation of fuzzy systems in FPGA to improve their response speed with respect to their operation in software.

The aforementioned works comprise different aspects of a common goal, to make driving land vehicles safer and more autonomous. For this research, first we collected the data with the GPS to observe both, the error of the measurements and its response time. Then, the hypothesis

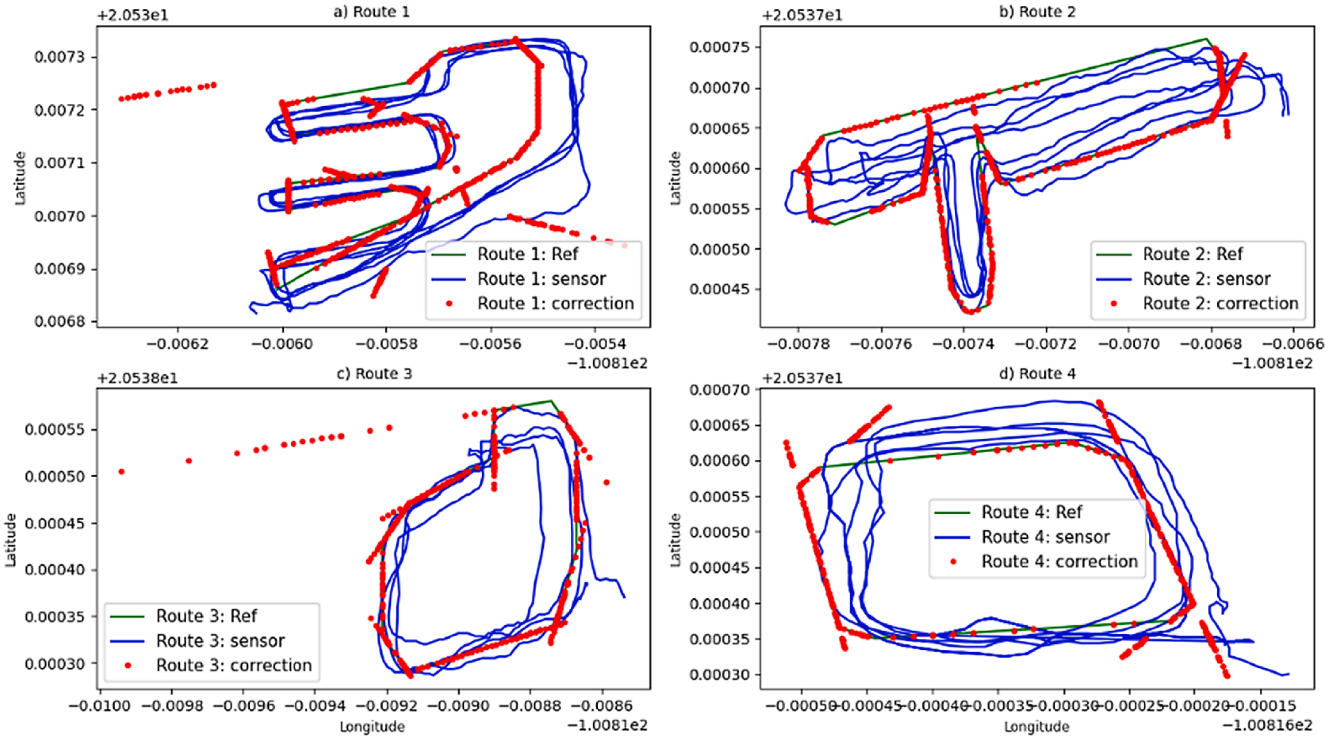

Fig. 2. Approximation of GPS points to the straight lines of the traveled routes.

Table 1
Training and validation data.

	Data Train	Data Test	Total
Route 1	751	250	1001
Route 2	645	215	860
Route 3	356	118	474
Route 4	412	102	514

was raised that it is possible to correct the error of the receiver using a fuzzy logic system and managing to implement said system in an FPGA to obtain a fast response speed in the correction of the absolute position. Next, the design and experimentation are described in Sections 2 and 3 respectively, as well as the analyses of the results obtained that validate the hypothesis raised in Section 4. Finally, we conclude in Section 5.

2. Materials and methods

2.1. Fuzzy absolute position correction system:

The latitude and longitude data supplied by the Pmod GPS sensor can be considered as points of a spherical plane (Earth sphere) that have a certain margin of error when paths, made up of lines and curves on this same sphere, are traveled, like the streets and highways of a city. The fuzzy position correction system was designed to carry out the task of approaching the points given by the Pmod GPS sensor to the lines that make up the roads of a city, see Fig. 1.

Distance equation between two points on a sphere [19]:

$$d = 2*r*\sin^{-1}\left(\sqrt{\sin^2\left(\frac{\varphi_2-\varphi_1}{2}\right)+\cos\varphi_1*\cos\varphi_2*\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right) \quad (1)$$

Considering sensor φ latitudes, λ lengths and r = 6371 [km].

Using the Haversines equation (1), it is possible to calculate the distance from each point given by the sensor to the straight lines that shape a specific highway (or route) of a city; after, the minimum distance is determined and the point is replaced in the equation of the closest line (2) to correct it, as shown in Fig. 2.

$$\lambda = m\varphi + b \quad (2)$$

with $m = \frac{\lambda_2-\lambda_1}{\varphi_2-\varphi_1}$ y $b = \lambda_1$

The fuzzy system was developed using Matlab's ANFIS (Adaptive Neuro Fuzzy Inference System) toolbox, which offers the convenience of training the system with a neural network and builds the logic of the fuzzy system from the Takagi-Sugeno-Kang (TSK) method. To achieve the training, corrected GPS data was obtained as described above, dividing this data block into training and validation sets. This data is presented in Table 1.

The design of the fuzzy correction system consists of two subsystems, the first one corrects the latitude data and the second one corrects the longitude data, due to the limitation of the ANFIS tool of training fuzzy systems with multiple inputs, but with only a single output. The diagram of the complete correction system is shown in Fig. 3.

Table 2 summarizes the structure (membership functions, number of inputs, and fuzzy rules) that makes up each fuzzy system for absolute position correction designed for this research; this information is relevant because of the consideration of each platform implementation, the number of carried out operations and the response time.

Fig. 4 shows the steps and information flow of the proposed fuzzy systems. Here, the inputs correspond to the data coming from the GPS sensor. In step 3, it corresponds to the internal structure of the TSK algorithm implemented by the MATLAB ANFIS toolbox for the latitude correction system with the subscript $i = 1, \ldots, 25$ for the fuzzy rules and the subscript $j = 1, \ldots, 5$ for the membership functions per input. On the other hand, in step 4 for the longitude correction, the fuzzy system contains 9 fuzzy rules and 3 membership functions for each input.

The output of the fuzzy system, obtained through the Evalfis function of Matlab, was compared with reference data taken manually from Google maps of the evaluated routes and the distance between them using equation (1), as seen in Fig. 5, to finally determine the effectiveness of the action of absolute position correction.

The training and validation of both fuzzy systems was performed using the Matlab toolbox ANFIS. This tool initializes the membership

Table 2
Fuzzy systems for absolute position correction.

Fuzzy System	# Inputs	# Membership functions for each input	# Fuzzy rules
Latitude	2	5 gaussian type 2	25
Longitude	2	3 gaussian type 2	9

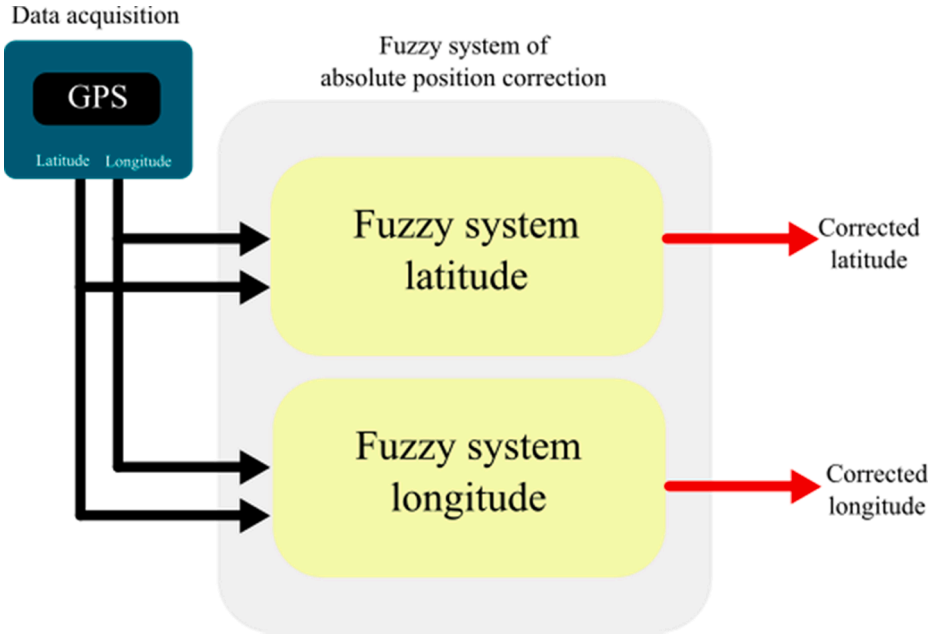

Fig. 3. General diagram of the absolute position correction fuzzy system.

- **Begin**

1- Load data:

- Define inputs vectors: *x* (latitude), *y* (longitude)

2- Define membership function (gaussian type 2):

- Function *gauss2mfp* with arguments*: input, mean1, sigma1, mean2, sigma2*
  - *idx1 = input <= mean1*
  - *idx2 = input > mean2*
  - *output[idx1] = gaussmf(input[idx1], mean1, sigma1)*
  - *output[idx2] = gaussmf(input[idx2], mean2, sigma2)*
  - return *output*

3- Define function fuzzy system of latitude:

-Function *fuzzylat* with arguments *x* and *y*

a) Fuzzification:

-F1j(x) = gauss2mfp(x, mean1, sigma1, mean2, sigma2)
-F2j(y)= gauss2mfp(y, mean1, sigma1, mean2, sigma2)

b) Inference (fuzzy rules):

$-wi = AndMethod\ (F1(x), F2(y))$

c) Rule aggregation:

-Rule output level is $zi = aix + biy + ci$
$- Aggregation = \sum_{i=1}^{25} zi * wi$

d) Defuzzification:

$- Zolat = \frac{\text{Aggregation}}{\sum_{i=1}^{25} wi}$

-Return *Zolat*

4- Define function fuzzy system of longitude:

-Function *fuzzylon* with arguments *x* and *y*

a) Fuzzification:

-F1j(x) = gauss2mfp(x, mean1, sigma1, mean2, sigma2)
-F2j(y)= gauss2mfp(y, mean1, sigma1, mean2, sigma2)

b) Inference (fuzzy rules):

$-wi = AndMethod\ (F1(x), F2(y))$

c) Rule aggregation:

-Rule output level is $zi = aix + biy + ci$
$- Aggregation = \sum_{i=1}^{9} zi * wi$

d) Defuzzification:

$- Zolon = \frac{Aggregation}{\sum_{i=1}^{9} wi}$

-Return *Zolon*

5- Define time function:

-function *time*
-run time= process_time()
-return: *run time*

6- Calculations

-while true:
-call functions:
run time= time(),
zolon=fuzzylon(x,y)
zolat=fuzzylat(x,y)

Fig. 4. Pseudocode of the fuzzy systems implementation.

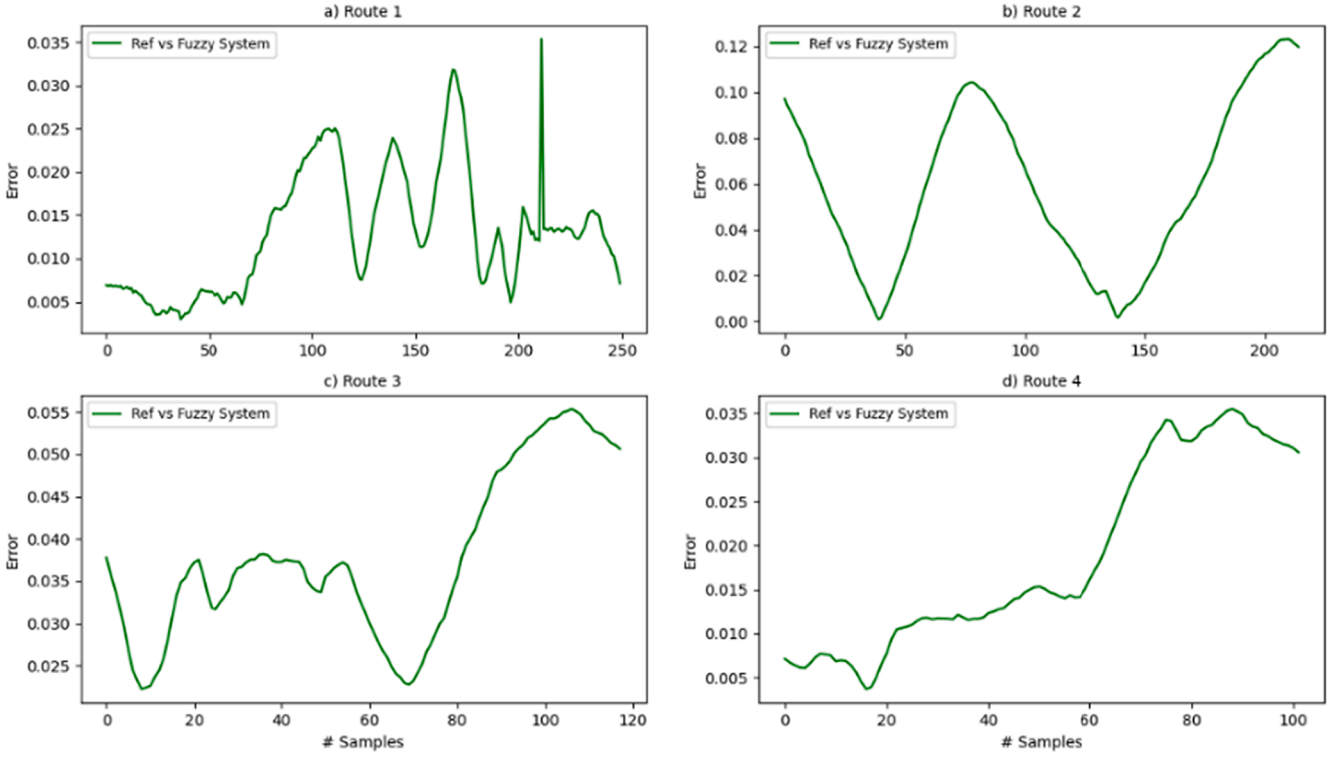

Fig. 5. Absolute position correction error (Reference vs Fuzzy system).

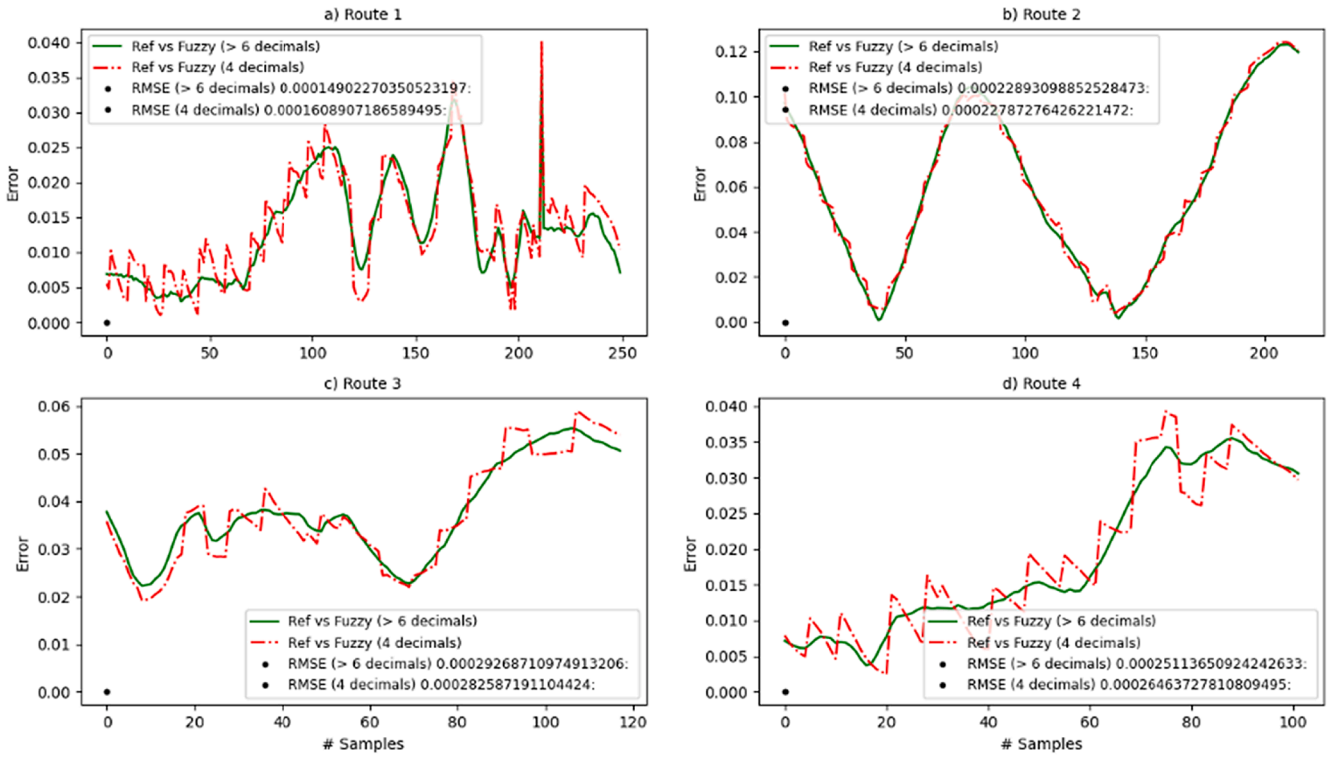

Fig. 6. Precision analysis of the fuzzy position correction system.

Table 3
RMSE of the fuzzy system output relative to the reference.

RMSE	Latitude (4 decimals)	Latitude (≥6 decimals)	Longitude (4 decimals)	Longitude (≥6 decimals)
Route_1	8.044535932947e-05	7.4511351752615e-05	8.044535932e-05	7.4511351752e-05
Route_2	11.39363821311e-05	11.446549426264e-05	11.39363821e-05	11.446549426e-05
Route_3	14.12935955522e-05	14.634355487456e-05	14.12935955e-05	14.634355487e-05
Route_4	13.23186390540e-05	12.556825462121e-05	13.23186390e-05	12.556825462e-05

functions with random values and uses a neural network to calibrate the parameters of both the membership functions of each input and the output (from the training data).

For the systems designed in this research, the membership functions are Gaussian type 2 and the parameters to be calibrated by the training network are their means and variances. The output is modelled with linear functions and the parameters to be optimized by the training network are the constants of these functions. In Fig. 4. in steps 3 a) and 3c) as well as 4 a) and 4c) the form of these parameters can be seen in a general way.

The training process is performed in software to take advantage of the ANFIS toolbox described above, then a precision analysis on the mathematical operations involved by the fuzzy systems is performed. From this analysis, an FPGA implementation is carried out using the parameters obtained in the software implementation.

2.2. *Precision Analysis:*

The Pmod GPS sensor has a precision range of 3 m to 7 m depending on the number of satellites that it manages to detect in a specific area of the earth plane; in numerical terms these precision variations represent changes in the number of decimals which the sensor delivers both in latitude and longitude and might be in a range from 8 to 19 decimal places.

For the precision analysis, the number of decimals in the data supplied by the sensor as input for the fuzzy system was reduced consecutively and then the output was recorded (using the Evalfis function of Matlab) to determine how much the correction error varies with respect

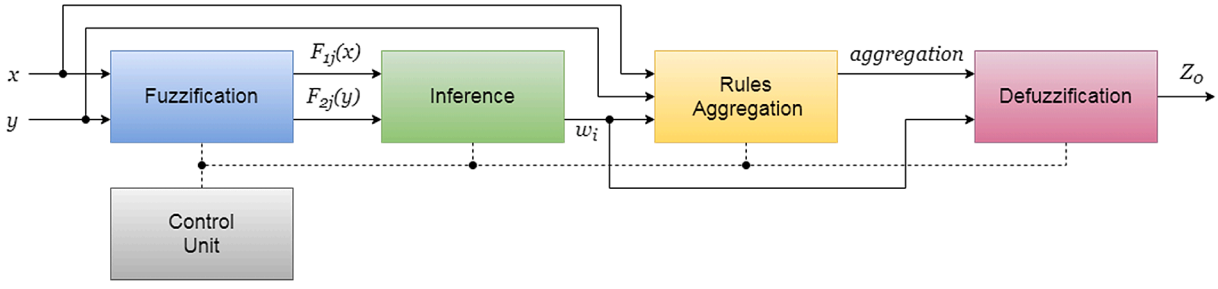

Fig. 7. FPGA-based architecture for latitude and longitude correction.

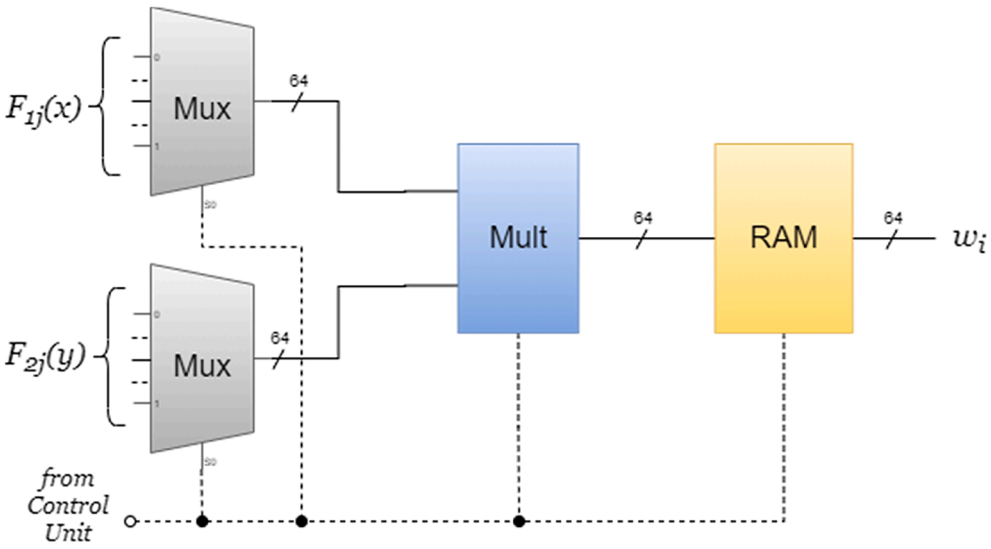

Fig 8. Inference module.

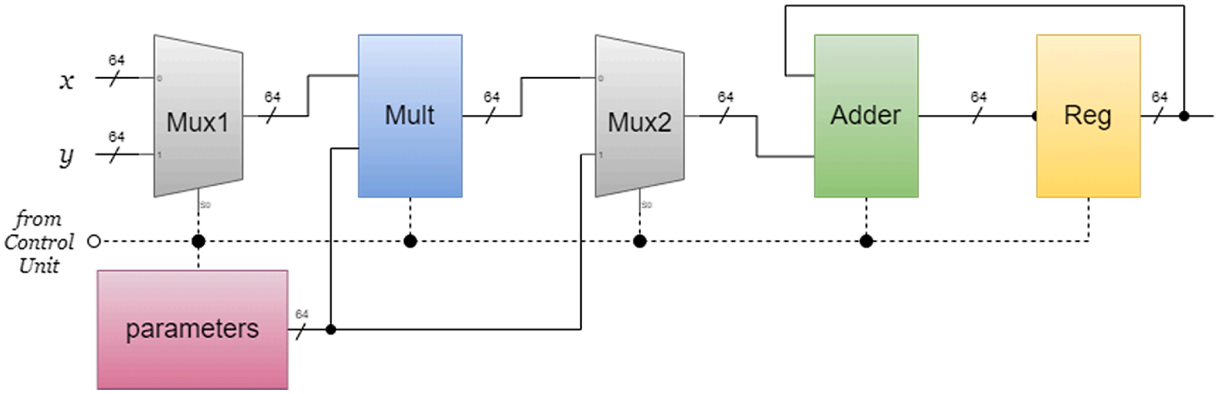

Fig. 9. Generation of membership functions.

to reference, with these changes. See Fig. 6.

The importance of this analysis relies in the possibility to find the minimum number of decimals to represent the sensor data and to carry out the operations required by the fuzzy systems on the FPGA. With this minimal expression of the data we can perform operations in a more efficient way without affecting the absolute position correction function performed by the fuzzy system, see Table 3.

2.3. FPGA implementation:

The response time of a location system based on GPS data must be fast when it is implemented in an autonomous land vehicle due to the risk that any delay or malfunction might represent for the security of the vehicle passengers. For that reason, the implementation of the fuzzy system of absolute position correction in FPGA is pertinent since the parallel processing of the data speeds up the function of the system.

Fig. 7 shows a block diagram of the FPGA architecture of the fuzzy subsystems proposed in Fig. 3. In this diagram, we observe the following modules: 1) Fuzzification; 2) Inference; 3) Rules Aggregation; 4) Defuzzification; and 5) Control Unit. The system receives as input the GPS data to be corrected: latitude (x) and longitude (y), generating as output the value of *Zo*. A description of each module is given below.

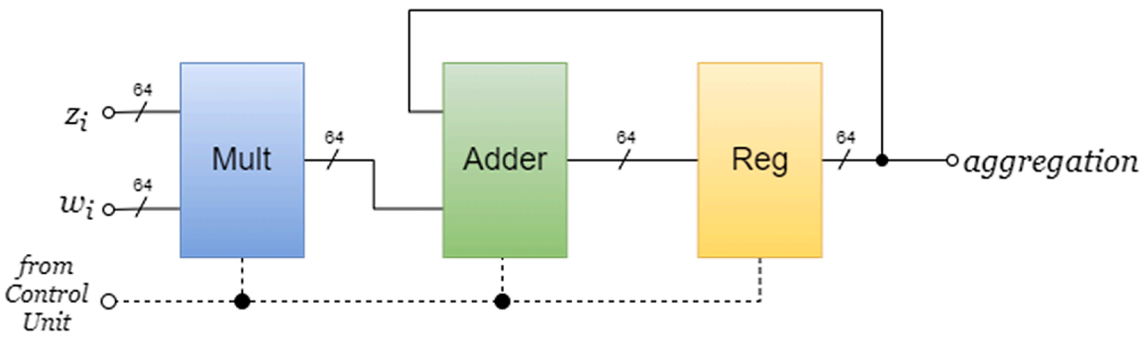

Fig. 10. Generation of the output *aggregation* of the Rules aggregation module.

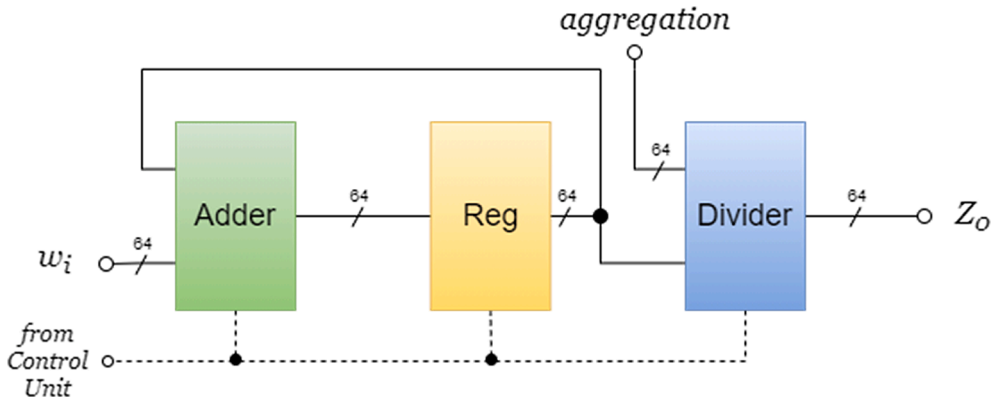

Fig. 11. Defuzzification module.

1) Fuzzification: This module implements the fuzzification operation for the external inputs, which calculates the degree of membership of the inputs to each of the membership functions defined by *j* (type 2 Gaussians). For the latitude and longitude correction systems *j* takes the values 5 and 3 respectively. On the hardware implementation, look-up tables (LUTs) were used to store the values of the membership functions.

2) Inference: This module receives as input the functions $F_{1j}(x)$ and $F_{2j}(y)$ generated by the Fuzzification module and delivers the weights (ω_i) to the output by applying the fuzzy rules (AND method) on all the combinations for the inputs (25 and 9 fuzzy rules for latitude and longitude respectively). For the fuzzy system, the product of the inputs was used as the AND method, and in the FPGA implementation, two multiplexers and a multiplier were used to generate it. The resulting outputs (ω_i) are temporarily stored in a RAM memory for later use in the following modules as shown in Fig. 8.

3) Rules Aggregation: It receives the $\times$ and *y* inputs, as well as the weights ω_i calculated by the Inference module, thus generating the aggregation output. This module is divided in two sub-modules: The first one, (Fig. 9) calculates the membership functions of the output, in our case, they correspond to linear functions of the form:

$$Z_i = a_i x + b_i y + c_i \tag{3}$$

where the coefficients $a_i, b_i yc_i$ are stored inside a RAM memory in the FPGA, the multiplexer Mux1 together with the multiplier perform the product of the inputs with their respective coefficient; while the Mux2, the adder and the register perform the cumulative sum of all the terms of the membership function to generate the output Z_i.

The second sub-module, shown in Fig. 10, receives the values z_i from the previous step and the weights ω_i, to generate the aggregation output by summing the products of the inputs, as shown in the following equation:

$$aggregation = \sum_{i=1}^{N} Z_i \omega_i \tag{4}$$

where *N* takes the values of 25 and 9 for the latitude and longitude correction systems respectively. As can be seen, a multiplexer, an adder and a bank of registers were used to generate the cumulative sum of the products $Z_i\omega_i$ to obtain the value for the aggregation output.

4) Defuzzification: The Defuzzification module is responsible for generating the output Z_0 from the input divided by the sum of the weights ω_i as follows:

$$Z_0 = \frac{aggregation}{\sum_{i=1}^{N} \omega_i} \tag{5}$$

where N takes the same values as in the Rules Aggregation module. For the hardware implementation of this module (Fig. 11), an adder and a register bank were used to perform the summation of the weights, as well as a divider whose function is to generate the output Z_0.

5) Control Unit: This module is responsible for generating the control signals necessary for the synchronisation of the different elements of the fuzzy system. For its implementation, two state machines were used to simplify its design. The first machine generates the control signals for the Fuzzification and Inference modules, as well as for the first Rules Aggregation submodule. On the other hand, the second state machine synchronises the summations of the second Rules Aggregation submodule and the Defuzzification module to perform the division of both and finally generate the output Z_0.

To implement the fuzzy systems a Terasic DE1-SoC development board was used, built around an Altera Cyclone V 5CSEMA5F31C6N FPGA. It has 32,070 adaptive logic modules (ALM), 87 DPS and 128,300 registers, in addition to having a1GB of SDRAM memory, and the programming language used in this work was VHDL. This is a low-level programming language for targeting implementations in reconfigurable hardware, such as FPGAs. To minimize rounding errors, double precision floating point (64 bits) was used for the representation of the data. Table 4 shows the hardware utilization, as well as the processing time for the fuzzy system. As we can observe, the system uses 18.64% of the FPGA logic, 6.39% of the registers and 55.17% of the DSP. Regarding the processing time, the subsystem that corrects the longitude took 8 μs to complete the calculations, while the subsystem that corrects the latitude, which is the heaviest computationally speaking, took 21.8 μs to perform the correction. Because both subsystems were implemented in parallel in the FPGA, the processing time corresponds to the time of the longitude correction subsystem.

2.4. Platform comparison for deployment

Both, the precision and the execution time of fuzzy systems were compared with their response given by Evalfis, the Matlab tool. Three hardware platforms were used for the implementation:

Table 4
Hardware utilization and processing time for the fuzzy system.

Fuzzy System	Logic utilization (ALMs)		Registers		DPS blocks		Time
Latitude	2984/32070	9.3%	4091/128300	3.19%	24/87	27.59%	21.8 μs
Longitude	2993/32070	9.33%	4103/128300	3.20%	24/87	27.59%	8 μs
Total	5977/32070	18.64%	8194/128300	6.39%	48/87	55.17%	21.8 μs

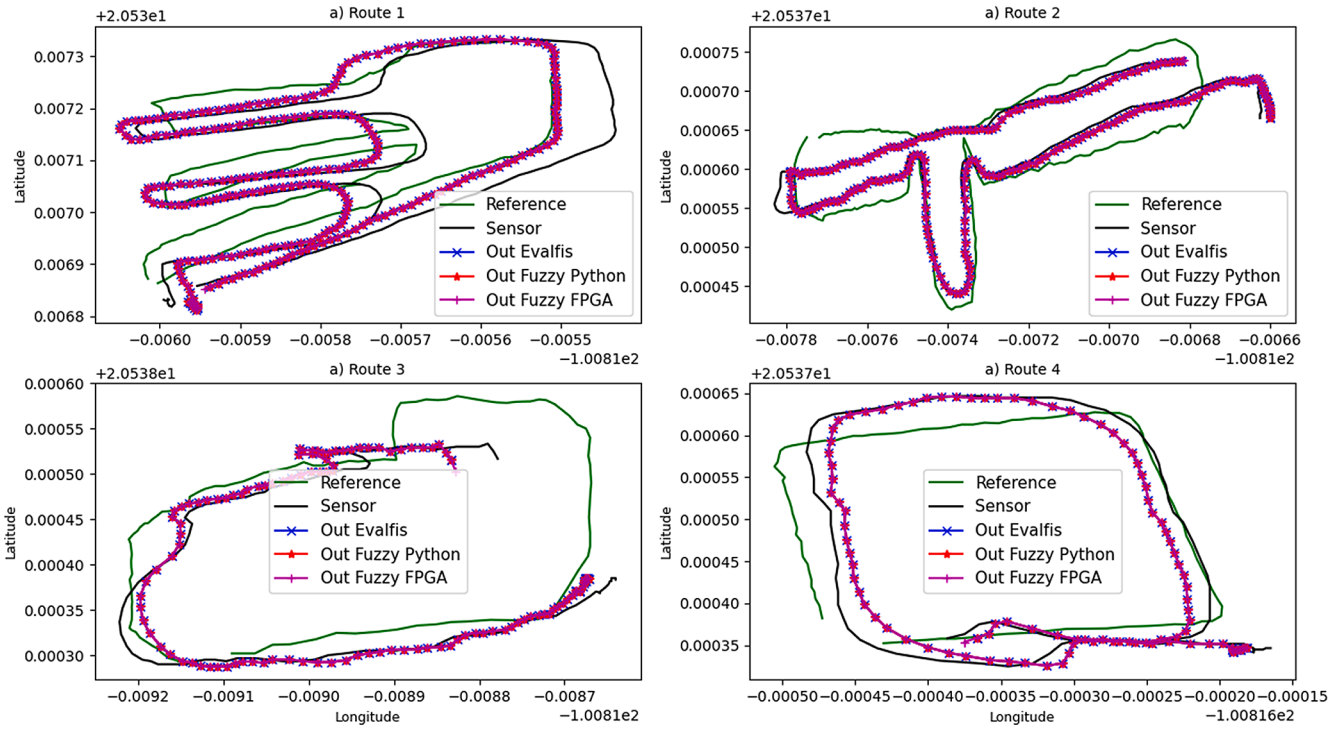

Fig. 12. Output of the fuzzy systems on different platforms.

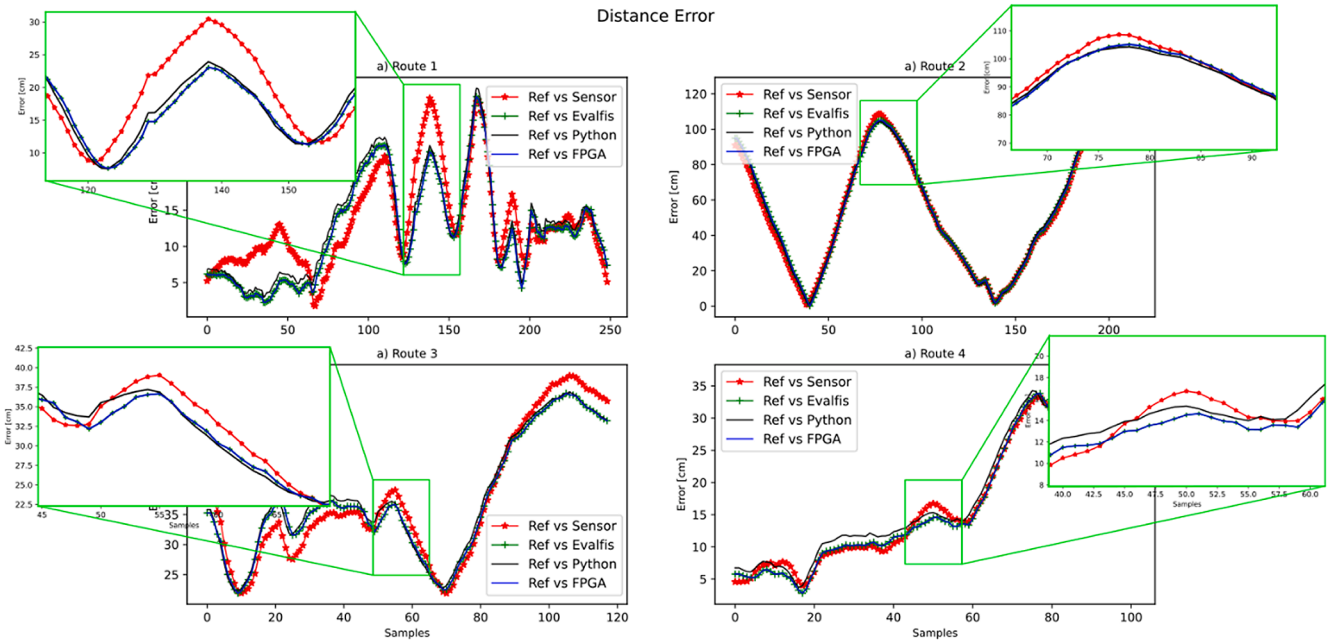

Fig. 13. Estimated error (distance in cm) from the fuzzy systems with respect to the reference.

a) A personal computer (PC) with Ubuntu 18.04.4 LTS 64-bit operating system, Intel Core i7-7740 @ 4.30 GHz x8 processor, 16 GB RAM and Nvidia 106 graphics card. The programming was performed using Python (High-level programming language).
b) A Raspberry pi 4 model B with Raspbian operating system, with Quad Core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHz processor, 4 GB RAM. Also, the programming was performed using Python (High-level programming language).
c) Finally, an FPGA implementation, an Altera Cyclone V 5CSEMA5F31C6N board was used. The programming was performed using VHDL (Low-level programming language).

In Fig. 12, the routes traveled are graphically observed from the data obtained by the fuzzy systems implemented in the three platforms arranged for comparison; there are differences in the decimals of these outputs that are graphically imperceptible but can be characterized numerically as shown in the following sections.

Despite the differences in decimals found in the output of fuzzy

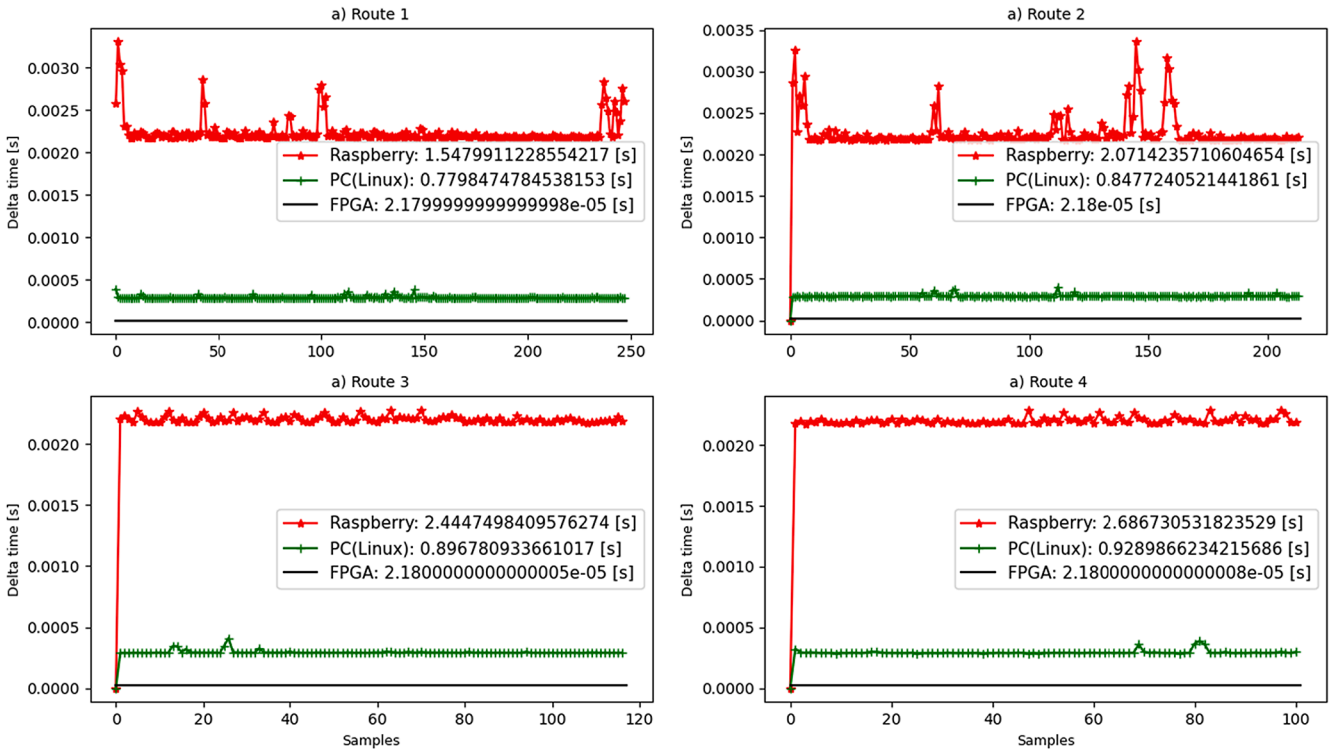

Fig. 14. Run time of fuzzy systems on each platform.

Table 5
Mean of the error (distance) of the fuzzy systems respect to the reference.

	Route_1	Route_2	Route_3	Route_4
Mean Error Ref vs Sensor [cm]	13.40122878687	57.51193967159	37.40618597547	18.21693033957
Mean Error Ref vs Evalfis [cm]	12.23377013779	58.29437245279	37.18648749282	17.67677087069
Mean Error Ref vs Python [cm]	12.94470407178	58.28557334455	37.64564916339	18.64302502601
Mean Error Ref vs FPGA [cm]	12.23377014250	58.29437245842	37.18648749909	17.67677087545

systems when implemented in each platform with respect to the reference and the output given by Matlab's evalfis function, these variations do not greatly affect the location of the point in the map (as seen in Fig. 12).

3. Results

This section presents the results obtained for the latitude and longitude correction systems.

Fig. 13 shows the estimated error of the different fuzzy systems with respect to the reference for the four proposed routes. Here, it can be observed that both the FPGA and the Python implementations performed well, since the error obtained is comparable (or similar) to that obtained with the Matlab evalfis tool.

In the implementation of fuzzy systems in Python to execute them, both on the Raspberry and on the PC, the scikit-fuzzy library was used, which presents some differences compared to the Matlab counterpart (Anfisedit); these differences also generate a change in the decimals of the diffuse exit. The output of the fuzzy implemented in Python is the same on the Raspberry and the PC, so the most notable differences between these two platforms is the execution time, as shown in Fig. 14.

Both fuzzy systems were implemented in the FPGA in parallel, making the slowest response time, which corresponds to the latitude correction system. As shown in Table 2, this fuzzy system has more membership functions, more fuzzy rules, therefore, more mathematical operations and a higher execution time compared to the other fuzzy system.

4. Discussion

The correction action of the fuzzy systems can be observed numerically in a specific way in the first two rows of Table 5. In the first row, the distance in centimeters that exists between the data received by the GPS sensor and the reference shows the error desirable to be decreased. The second exposes the correction made by the fuzzy systems comparing their output obtained by the Evalfis function (from Matlab) to the reference, decreasing that distance in most of the traveled routes of the test. This row serves as a basis for comparing the accuracy of the systems when implemented in each of the hardware platforms.

In the third row of Table 5 it is clearly exposed, how the variation in the decimals resulted from the mathematical operations carried out within the fuzzy systems (implemented in Python), directly affects its output, and on average making it move even further away from the reference than the pure data from the GPS sensor. As can be seen in Fig. 12, despite this variation when locating the latitude and longitude point on the map, it continues to fall where it should be after being modified by fuzzy systems, that is, the precision loss in the implementation of fuzzy systems in Python is not representative enough to deform the path traveled in the test and obscure the correction action of fuzzy systems.

Fig. 13 graphically presents the error (distance) of the fuzzy systems output with respect to the reference and the same comparison of the mentioned output on each hardware platform. Contrary to the implementation in Python, the data obtained from the systems implemented in the FPGA show great similarity with the output of the Evalfis function

Table 6
Error (distance) between the implementation in Python and FPGA regarding the response of Evalfis.

	Route_1	Route_2	Route_3	Route_4
Error Python vs Evalfis [cm]	0.845778043	1.07005758	0.676937288	1.000264318
Error FPGA vs Evalfis [cm]	4.110605614e-08	5.119604309e-08	5.17049727e-08	4.235469112-e08

Table 7
Execution time on each platform.

	Route_1	Route_2	Route_3	Route_4
PC(Linux) [s]	0.77984748	0.84772405	0.89678093	0.92898662
Raspberry pi 4 [s]	1.54799112	2.07142357	2.44474984	2.68673053
FPGA [s]	21.8e-06	21.8e-06	21.8e-06	21.8e-06

(from Matlab), a situation that is confirmed in the last row of Table 5, indicating a smaller modification of the decimal places in the internal mathematical operations of fuzzy systems within the FPGA.

In the Table 6, a numerical comparison of the difference in distance of the fuzzy output given by the Evalfis function of Matlab and the data delivered by the Python and FPGA implementations respectively is made. The inherent error in the Python implementation is just over 1.07 cm, making this tolerance margin not perceptible from perspective of locating the point on the map. Also, as evidenced in Fig. 6, a much more considerable variation of decimals, which have the output of the fuzzy systems, is needed to damage the absolute position correction of the fuzzy ones.

The answer given by the FPGA shows as the maximum difference the distance of $51,7x10^{-9}$cm with the data given by the Evalfis function of Matlab. This error is practically negligible in terms of the location of the GPS points on the map and of the loss of precision as an effect of the implementation of fuzzy systems in this hardware platform.

The execution time of fuzzy systems, in addition to precision, becomes very important for the application, since a rapid response of the system is needed to locate an autonomous vehicle in space with the least possible error and from there follow a route. In Fig. 14, a comparative graph of the execution time is presented for each hardware platform. The Raspberry pi 4 turns out to be the slowest due to the sequential processing of the two fuzzy ones and the reduction of computational resources in tasks related to the operation of the Rasbian operating system, which at certain moments slows down even more the operation of the application. The behavior of the application on the PC is like that of the Raspberry since the execution time is variable, as it is implemented sequentially.

As established in Table 7, the average response time in each path is below 1 s for the PC and greater than 2 s for the Raspberry pi 4. As the FPGA has dedicated hardware exclusively for parallel operation, it has a much rapid and constant response time for all 21.8 μs samples.

5. Conclusions

Fuzzy absolute position correction systems aim to reduce the error of GPS sensor measurements relative to the reference as much as possible. That difference cannot be reduced when the sensor has a fault or passes through an area where there are many obstacles between the GPS satellites and the receiving sensor.

Autonomous vehicles need, as a first step, to start a movement to locate themselves in space and GPS sensors offer this information; However, the pure data from the sensor is not enough to have reliable information and that is why a refinement process must exist, but also, this stage must be as fast as possible so that the vehicle's navigation algorithm can decide. Fuzzy logic offers a solution for correcting GPS data and, in combination with the FPGA, a fast and reliable response of the absolute position of an autonomous land vehicle can be achieved.

The comparisons between the different hardware platforms evaluated in this article present several alternatives for the implementation of an artificial intelligence system and GPS-based absolute position estimation, considering the computational cost, the response speed and, implicitly, the monetary cost of each case application. Considering the speed of the system, the FPGA fully meets the requirements of the application, as well as the computational cost of processing the two fuzzy systems in parallel. Albeit the large number of arithmetic operations required for the fuzzy systems, it was possible to optimize the available resources in the FPGA. Also, the FPGA delivers high precision results because the design was implemented in 64 bits floating point.

CRediT authorship contribution statement

Pedro J. Correa-Caicedo: Conceptualization, Methodology, Software, Data curation, Writing - original draft, Validation, Writing - review & editing. **Alejandro I. Barranco-Gutiérrez:** Conceptualization, Methodology, Writing - original draft, Supervision, Writing - review & editing. **Erick I. Guerra-Hernandez:** Software, Validation. **Patricia Batres-Mendoza:** Software, Validation, Writing - review & editing. **Jose A. Padilla-Medina:** Visualization, Investigation. **Horacio Rostro-González:** Conceptualization, Methodology, Writing - original draft, Supervision, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors greatly appreciate the support of TecNM, CONACyT. This work was partially funded by the CONACYT grant FC-2016-1961.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References:

[1] R. Woo, E. Yang, A fuzzy-innovation-based adaptive Kalman filter for enhanced vehicle positioning in dense urban environments, Sensors (Basel) 19 (5) (2019 Mar) 1142, https://dx.doi.org/10.3390%2Fs19051142.
[2] J. An, Y. Yu. Fuzzy-Based Hybrid Location Algorithm for Vehicle Position in VANETs via Fuzzy Kalman Filtering Approach. Advances in Fuzzy Systems. Volume 2019, Article ID 5142937. https://doi.org/10.1155/2019/5142937.
[3] M. Tehrani, N. Nariman. Adaptive fuzzy hybrid unscented/H-infinity filter for state estimation of nonlinear dynamics problems. Transactions of the Institute of Measurement and Control, Vol 41, Issue 6, 2019. https://doi.org/10.1177/0142331218787607.
[4] C. Shen, et al., Seamless GPS/Inertial Navigation System Based on Self-Learning Square-Root Cubature Kalman Filter, IEEE Trans Ind Elec 68 (1) (Jan. 2021) 499–508, https://doi.org/10.1109/TIE.2020.2967671.
[5] R. Gonçalves, S. Givigi. FPGA-Based Design Optimization in Autonomous Robot Systems for Inspection of Civil Infrastructure. IEEE Systems Journal (Volume: 14, Issue: 2, June 2020), 2961 – 2964. https://doi.org/10.1109/JSYST.2019.2960309.
[6] A. Ghorbel, N. Ben, Design of a flexible reconfigurable mobile robot localization system using FPGA technology", Computer and Embedded System Laboratory, SN Appl Sci volumen 2 (2020) 1183, https://doi.org/10.1007/s42452-020-2960-4.

[7] M. El, G. Hamza. FPGA implementation of tracking phase of the GPS receiver using XSG. Journal of Communications Vol. 15, No. 4, April 2020. https://doi.org /10.24425/ijet.2019.130257.
[8] L. Segers, A. Braeken. Optimizations for FPGA-based ultrasound multiple-access spread spectrum ranging. Journal of Sensors. Volume 2020, Article ID 4697345. https://doi.org/10.1155/2020/4697345.
[9] T. Li, Y. Ma, FPGA implementation of real-time pedestrian detection using normalization-based validation of adaptive features clustering, IEEE Trans Vehicular Tech 69, Issue: 9 (2020) Sept, https://doi.org/10.1109/TVT.2020.2976958.
[10] W. Shia, M. Baker, Algorithm and hardware implementation for visual perception system in autonomous vehicle: A survey, Integration 59 (September 2017) 148–156, https://doi.org/10.1016/j.vlsi.2017.07.007.
[11] S. Dua, T. Huang, FPGA based acceleration of game theory algorithm in edge computing for autonomous driving, J Sys Arc 93 (February 2019) 33–39, https://doi.org/10.1016/j.sysarc.2018.12.009.
[12] Z. Slanina, V. Kasik, GPS Synchronisation for FPGA Devices, IFAC Proce Vol 45 (7) (2012) 337–340, https://doi.org/10.3182/20120523-3-CZ-3015.00064.
[13] X. Yu, Y. Sun, Design and realization of synchronization circuit for GPS software receiver based on FPGA, J Sys Eng Elec 21, Issue: 1 (2010) Feb, https://doi.org/10.3969/j.issn.1004-4132.2010.01.004.
[14] Mata-Carballeira Ó, Gutiérrez-Zaballa J. An FPGA-Based Neuro-Fuzzy Sensor for Personalized Driving Assistance. Sensors (Basel). 2019;19(18):4011. Published 2019 Sep 17. doi:10.3390/s19184011.
[15] H. Huang, Fusion of Modified Bat Algorithm Soft Computing and Dynamic Model Hard Computing to Online Self-Adaptive Fuzzy Control of Autonomous Mobile Robots, IEEE Trans Ind Infor 12 (3) (June 2016) 972–979, https://doi.org/10.1109/TII.2016.2542206.
[16] Mohieddin Moradi, Mehdi Ehsanian,Design of an FPGA based DPLL with fuzzy logic controllable loop filters with application customization capability, AEU - International Journal of Electronics and Communications, Volume 97, 2018, Pages 54-62, ISSN 1434-8411, https://doi.org/10.1016/j.aeue.2018.09.026.
[17] Y. Zhou, Q. Fan, M. Liu, J. Ren, T. Zhou and Y. Su, "Design of Force-to-Rebalanced System with Adaptive Fuzzy-PID Controller for N=3 MEMS Disk Gyroscope," in IEEE Sensors Journal, doi: 10.1109/JSEN.2021.3068152.
[18] J. Pérez Fernández, Vargas M. Alcázar, Low-Cost FPGA-Based Electronic Control Unit for Vehicle Control Systems, Sensors (Basel) (2019;19(8):1834.), https://doi.org/10.3390/s19081834. Published 2019 Apr 17.
[19] C. Song, J. Kim. FPGA Implementation of Fuzzy Interpreted Petri Net. IEEE Access, 61442 – 6145, 2020. https://doi.org/10.1109/ACCESS.2020.2983276.
[20] N. Farah, et al., A Novel Self-Tuning Fuzzy Logic Controller Based Induction Motor Drive System: An Experimental Approach, IEEE Access 7 (2019) 68172–68184, https://doi.org/10.1109/ACCESS.2019.2916087.
[21] Q. Yang, L. Sun, A fuzzy complementary Kalman filter based on visual and IMU data for UAV landing, Optik - Int J Light Electron Optics 17 (2018), https://doi.org/10.1016/j.ijleo.2018.08.011.