Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXVII, págs. 609-618 (artículo completo)
A. Cardona, L. Garelli, J.M. Gimenez, P.A. Kler, S. Márquez Damián, M.A. Storti (Eds.)
Santa Fe, 5-7 Noviembre 2019

UAV AUTONOMOUS NAVIGATION BY DATA FUSION AND FPGA

Gerson da Penha Neto[a], **Haroldo Fraga de Campos Velho**[a] **and Elcio Hideiti Shiguemori**[b]

[a]*Laboratório de Computação e Matemática Aplicada (LABAC), Institudo Nacional de Pesquisas Espaciais (INPE), São José dos Campos, SP, Brasil, http://www.lac.inpe.br*

[b]*Instituto de Estudos Avançado (IEAv),Departamento de Ciência e Tecnologia Aeroespacial (DCTA), s, São José dos Campos, SP, Brasil, http://www.ieav.cta.br/*

Keywords: Unmanned aerial vehicles, FPGA: Field Programmable Gate Array, autonomous navigation, self-configuring neural network.

Abstract. Currently, the use of unmanned aerial vehicles (UAV), also known as *drones*, is increasing. The applications are in several areas such as engineering projects, agriculture, livestock, monitoring, and rescue. One of the main reasons to use UAV is its lower cost when compared to manned aircraft. The flight of a UAV can be done remotely or autonomously. For the autonomous navigation, a Global Navigation Satellite System (GNSS) is usually applied. However, a GNSS system can suffer natural or human interference, becoming the research for alternatives strategies a hot topic in this field. An approach to carry out the autonomous navigation without use of GNSS signal is to estimate the UAV position by using data fusion combining different sensors. A solution for autonomous navigation is presented applying inertial sensor and image processing, both are employed to estimate the drone position. The data fusion process is carried out by a computational intelligence procedure. Two self-configuring ANNs are employed here: for image edge extraction, and an operator for data fusion. A hybrid computer architecture is employed to implement the solution with standard CPU and FPGA (Field Programmable Gate Array).

1 INTRODUCTION

Unmanned aerial vehicles (UAVs), also known as remotely piloted aircraft or *drones*, and their applications are highlighted in recent years (Goltz et al., 2011; Valavanis and Vachtsevanos, 2014; Fiori et al., 2017). UAV can be applied for areas such as agriculture and livestock (Eltner et al., 2014), rescue operations (Goncalves and Renato, 2015), land mapping (James et al., 2017), and environmental and climate monitoring (Dash et al., 2017). A solid growth in the use of UAV is expected in the future, due to the lower cost of manufacturing and operating in comparison to traditional manned aircraft.

One topic of interest is to develop technologies for the control and navigation of UAVs. The UAV navigation can be performed in a remote way or autonomously (Braga et al., 2015).

For remote navigation, it is necessary a navigation protocol, implemented on the ground station, usually controlled by a radio control. The management and communication of the aircraft can be done by a human pilot or software. Autonomous navigation does not require a human pilot, ground station, or radio control. For this navigation type, the control of the aircraft is done by an embedded computer system. The computer selects which signal should to be applied from embedded sensors, for example, from the Inertial Navigation System (INS) or/and from the Global Satellite Navigation System (GNSS).

There are some issues associated with remote navigation. Radio control communication has a range restriction, where many UAV systems use IEEE 802.11-based wireless technologies. In addition, interference on the radio signal implies in vulnerability associated to natural phenomena or not (Groves, 2015). This work investigates technologies for autonomous navigation in the presence of failures or absence of the GNSS signal.

2 MULTI-SENSOR DATA FUSION AND AUTOCONFIGURED ARTIFICIAL NEURAL NETWORK

Data fusion from multi-sensors is one approach for autonomous navigation. For such approach, the position of the aircraft during navigation is estimated using the fusion of information obtained from the embedded sensors. In the situation where the GNSS signal is not available, data fusion proposes to find the UAV position by combining different strategies, with computer vision techniques and INS sensor (Conte and Doherty, 2009).

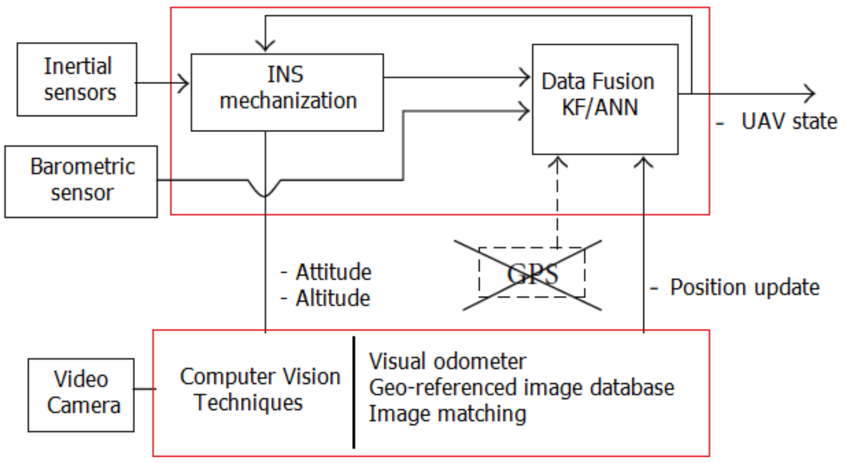


Figure 1: autonomous navigation by multi-sensor data fusion.

One formulation for computer vision is based on edge extraction from the a georeferenced image, hereafter called *reference image*, and the image obtained by the UAV camera. The convolution operation from segmented images is a scheme to estimate UAV position Conte and Doherty (2009). Conte and Doherty Conte and Doherty (2009) have used Kalman filter to determine the aircraft position, employing the INS data and computer vision. Figure 1 shows a schematic for estimating the position of the aircraft; using data fusion.

In the present work, a new data fuser is proposed, where Kalman filter is replaced by an artificial neural network (ANN). The ANN used here is the self-configured multilayer perceptron, and the back-propagation algorithm is employed to calculate the connection weights – the *learning phase*.

For image edge extraction, the patterns shown in Figure 2 are applied to identify edges and non-edges image pixels Braga et al. (2015, 2016). Another self-configured ANN was defined for image edge extraction.

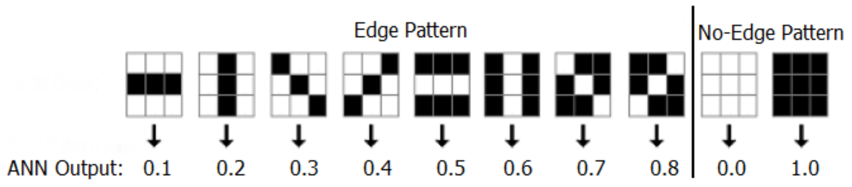


Figure 2: Patterns applied in artificial neural network training, applied in edge extraction.

The architecture for the both cited ANNs is computed as an optimization problem. The objective is to find optimal values for the number of hidden layers, number of neurons for each hidden layer, the learning ratio and momentum parameter, and the type of activation function. For finding the optimal solution, the meta-euristic called Multiple Particle Collision Algorithm (MPCA) was applied (Anochi and Velho, 2014; Anochi et al., 2015). The cost function ϕ to be minimized is a combination of two factors: a metric for the ANN complexity – the first term in the right hand side in the below equation, and the quadratic difference between the desired values and the ANN output:

$$\phi = \left[\theta_1 e^{x^2} + \theta_2 y + 1\right] \left(\frac{\rho_1 E_t + \rho_2 E_g}{\rho_1 + \rho_2}\right) \tag{1}$$

where x is the number of neurons; y corresponds to the number of epochs until the convergence during the learning phase, θ_1 and θ_2 are adjustment parameters to calculate the complexity quantification of the neural network; and ρ_1 and ρ_2 relates to the balance between learning (E_t) and generalization (E_g) errors.

3 REAL FLIGHT EXPERIMENT

The flight experiment was carried out at the Linköping University, Sweden. The Aircraft used was the Yamaha Rmax UAV.

During the experiment, one hundred and thirty-nine types of real-time data were captured. Data were collected from embedded sensors, such as compass, barometer, GPS, INS, gyroscope and position of the servo motors, among others. Here, the INS (Accelerometers, Gyroscopes), barometer and video camera were used, Table 1 presents a description of the data. Data fusion was performed by KF – see Figure 1.

There is a difference in the acquisition rate between the sensors. The KF processing rate is 50 Hz. Therefore, the KF was applied only when the lowest acquisition rate sensor provides

Sensor	acquisition rate	Resolution
Accelerometers	66 Hz	1 mG
Gyroscopes	200 Hz	0.1 deg/s
Barometer	40 Hz	0.1 m
Camera	25 Hz	384×288 pixels

Table 1: Some parameters associated to sensors used in the autonomous navigation algorithm.

a new data. Taking this into account, the activation of the neural network is made only when there is a change in the data obtained from the sensor with the lowest acquisition rate. In other words, for some data fusion process, data from the faster sensors are not considered.

3.1 Neural Fuser

As mentioned before, the KF estimation was replaced by a self-configured ANN. The MPCA algorithm seeks the best architecture for an ANN, through an optimization problem. One of the interests was to implement the ANN in a dedicated hardware, with FPGA (Field Programmable Gate Array). Hence the lower the network architecture the better it will be, in other words the fewer layers and neurons per layer it has, the less costly the implementation will be in FPGA and the less space will be required for ANN design.

Parameter	value computed
Neurons in the input layer	12
Number of hidden layers	2
Neurons in the hidden layer	10
Neurons in the output layer	2
Learning rate	0.7
Rate of momentum	0.52
Activation Function	Sigmoid

Table 2: ANN architecture to emulate KF.

The MPCA algorithm was configured to find the lowest architecture with the best performance; For this two experiments were performed. In the first experiment the configuration of the MPCA algorithm was defined so that the ANN architecture could have only one hidden layer. The number of neurons for the hidden layer was also modified and varied; Architectures with three, five, seven, ten and twenty neurons were tested in the hidden layer. The second experiment was similar to the first one, but the configuration of the MPCA algorithm was defined so that ANN has two hidden layers; In this experiment the same variation occurred in the number of neurons per hidden layer. In both experiments the autoconfigured ANN output layer has two neurons that provide the estimation of latitude and longitude respectively.

The total of 16,595 temporal data were used for self-configured ANN training in the two experiments. The data set was separated as follows: seventy percent were separated for the training phase, ten percent separated for the generalization phase and twenty percent for the validation phase. Each temporal data defined a vector with twelve values that are latitude and longitude, estimated by the image processing, data of the inertial sensor (Accelerometers, Gyroscopes) and altitude obtained by a barometer sensor; Same data used in data fusion performed by KF.

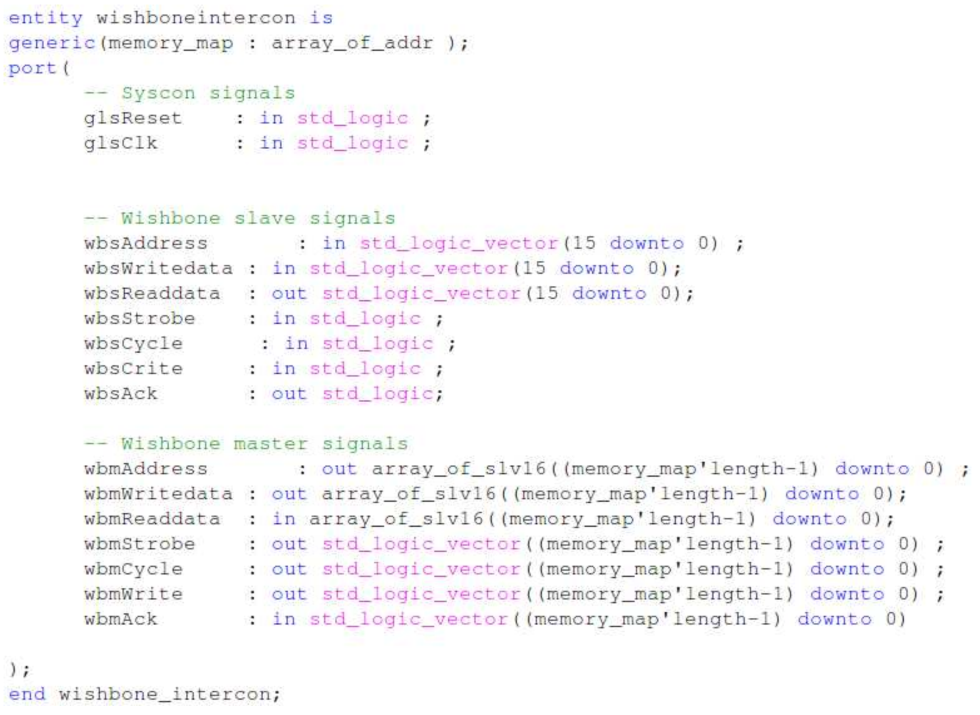
The results show that the best architecture for the autoconfigured ANN to emulate the KF was that it had two hidden layers with ten neurons per hidden layer. This architecture obtained the smallest error in the training, generalization and validation phases. Table 3 presents a summary of the error for each of the experiments performed and Table 2 presents a summary of the architecture obtained by the MPCA.

Number of hidden layers	Number of neurons per layer	Error during the training phase
1	3	$1.57755221369082 \times 10^{-4}$
1	5	$7.540736119348386 \times 10^{-5}$
1	7	$7.601787260976976 \times 10^{-5}$
1	10	$7.681266996990718 \times 10^{-5}$
1	20	$7.806631084675314 \times 10^{-5}$
2	3	$7.113050064122057 \times 10^{-5}$
2	5	$6.12196424759292 \times 10^{-5}$
2	7	$6.007679928803779 \times 10^{-5}$
2	10	$5.993148180318353 \times 10^{-5}$
2	20	$6.025567521060393 \times 10^{-5}$

Table 3: Training error for ANN training self-configuring with only one hidden string and for training with two hidden layers; With variation in the amount of neurons per layer.

4 IMPLEMENTATION ON FPGA

One of the our objectives is to obtain dedicated hardware for the neural fuser to be onbard in the aircraft. The VDHL codification was used to configure the FPGA as a neural fuser. The used FPGA is from the family Xilinx spartan-6, and it is linked to a raspberry-pi type computer. This computer is a small device with low cost.



```
entity wishboneintercon is
generic(memory_map : array_of_addr );
port(
      -- Syscon signals
      glsReset     : in std_logic ;
      glsClk       : in std_logic ;


      -- Wishbone slave signals
      wbsAddress       : in std_logic_vector(15 downto 0) ;
      wbsWritedata : in std_logic_vector(15 downto 0);
      wbsReaddata  : out std_logic_vector(15 downto 0);
      wbsStrobe    : in std_logic ;
      wbsCycle      : in std_logic ;
      wbsCrite     : in std_logic ;
      wbsAck       : out std_logic;

      -- Wishbone master signals
      wbmAddress       : out array_of_slv16((memory_map'length-1) downto 0) ;
      wbmWritedata : out array_of_slv16((memory_map'length-1) downto 0);
      wbmReaddata  : in array_of_slv16((memory_map'length-1) downto 0);
      wbmStrobe    : out std_logic_vector((memory_map'length-1) downto 0) ;
      wbmCycle     : out std_logic_vector((memory_map'length-1) downto 0) ;
      wbmWrite     : out std_logic_vector((memory_map'length-1) downto 0) ;
      wbmAck       : in std_logic_vector((memory_map'length-1) downto 0)

);
end wishbone_intercon;
```

Figure 3: Component responsible for sending and reading data on the FPGA board.

The communication between the raspberry-pi computer and the FPGA is serial, with the bus capacity of 16 bits. All data are normalized into interval $[0, 1]$. In order to reduce the processing in the FPGA, data is scaled to an integer number, where the scale is defined according to the required precision. This *new* number has more than 16 bits, so the number is separated into four partitions, with 16 bits for each partition. After that, all partitions are sent to the FPGA, and grouped inside the co-processor.

The control of sending and reading data is done by a component called `wishboneIntercon`. The `wishboneIntercon` is composed of signals that receive information of type `std_logic` and `std_logic_vector`. The latter directive is responsible for indexing each FPGA component address ensuring that a given data is sent or read from the correct component. Figure 3 shows the VHDL description of the `wishboneIntercon` component. The input signals for sending data are `wbsAddress, wbsWritedata`. The output signal `wbsReaddata` is used to read data. The `wbsStrobe, wbsCycle, wbsWrite,` and `wbsAck` are flags. The `wbmAddress, wbmWritedata`, and `wbmReaddata` signals are used for sending and reading the internal components in the FPGA. The flags for internal data traffic, are the `wbmStrobe, wbmCycle, wbmWrite`, and `wbmAck` signals.

For each data sent to the FPGA, four registers are used, because each register receives a 16 bits word. The input vector for the ANN has twelve inputs. Therefore, fifty six registers were used to transfer the data to perform the fusion by the ANN. After sending the data, the information needs to be reassembled, and this action is done by concatenating the 16 bits partitions from the transferred values into a value of 64 bits.

The same representation in 64 bits is applied to the values of weights and bias, and equal or very close values are shared, i.e., if a same weight value is used for more than one neuron, a VHDL constant is created to be used more than once. This strategy is also applied to the activation function case. The activation function is represented by a lookup table instruction.

5 FINAL REMARKS

The KF was emulated by the neural fuser implemented on FPGA. In the neural fuser, A neural network with two hidden layers and ten neurons for each hidden layer was employed, because this architecture obtained the smallest training error.

Figures 4 and 5 show the results of self-configured ANN activation for the second experiment; Where the network has two hidden layers. The latitude and longitude estimates made by the neural fuser are shown and the data presented are for the validation set corresponding to twenty percent of the total data obtained during the flight. The results for each variation in the number of neurons in the hidden layers are shown. From the presented result it is possible to see that the neural fuser can emulate the KF.

There is an error in the result of the neural fuser estimation; A distance between the measurements obtained with the neural fuser and the measurements obtained by KF. This error calculated in cm is close to 0.6×10^{-4} cm. This means that even if there is a difference it is sub-metric and does not compromise the neural fuser result. Figure 6 shows the error in cm for the validation set - same set of data shown in Figures 4 and 5; For each measure used in the activation of the neural fusion, the difference in cm between the measures estimated by the KF and the measures estimated by the neural fuser was calculated. The smaller the number of neurons in the hidden layers the greater the error in cm between the estimates made by the neural fuser and the estimates made by KF; The smallest distances are for when the hidden layers have the number of neurons between ten or twenty and the biggest distances are when the hidden layers have between three and five neurons.

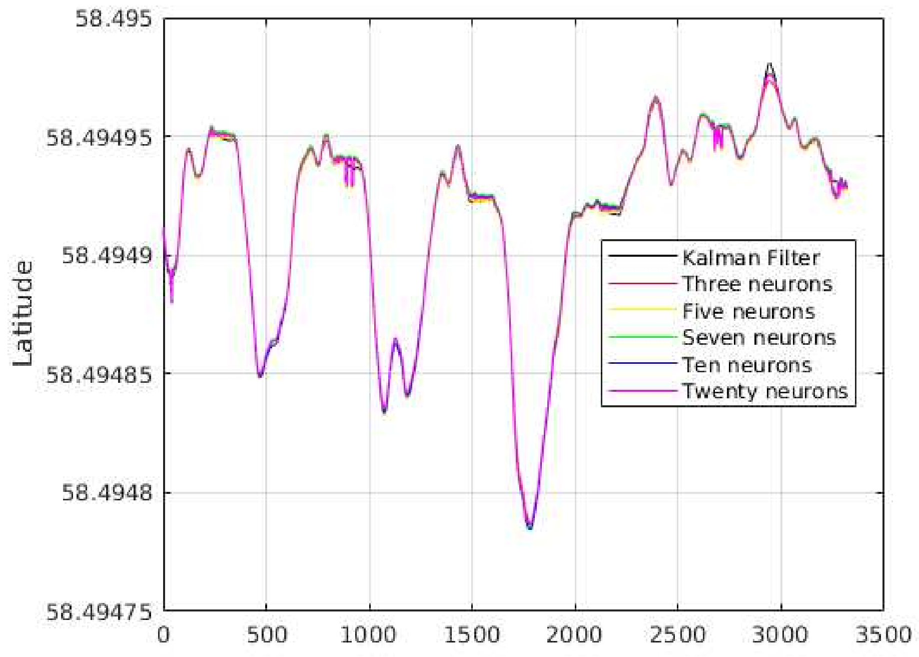


Figure 4: Latitude for two hidden layers.

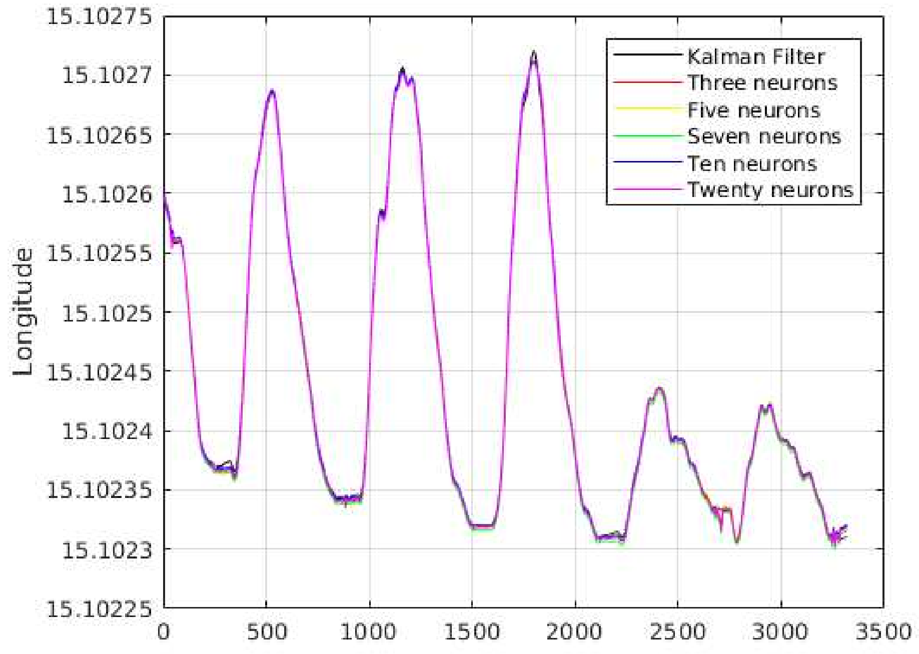


Figure 5: Longitude for two hidden layers.

The algorithimic complexity of the neural fuser is simpler than the KF, since the neural fuser does not require information from the dynamics of the aircraft. The neural fuser is just a non-linear mapping between the inputs and desired outputs, fully formulated by data driven; This means that the neural fuser can be an example of other ways of advertising. Figures 7 and 8

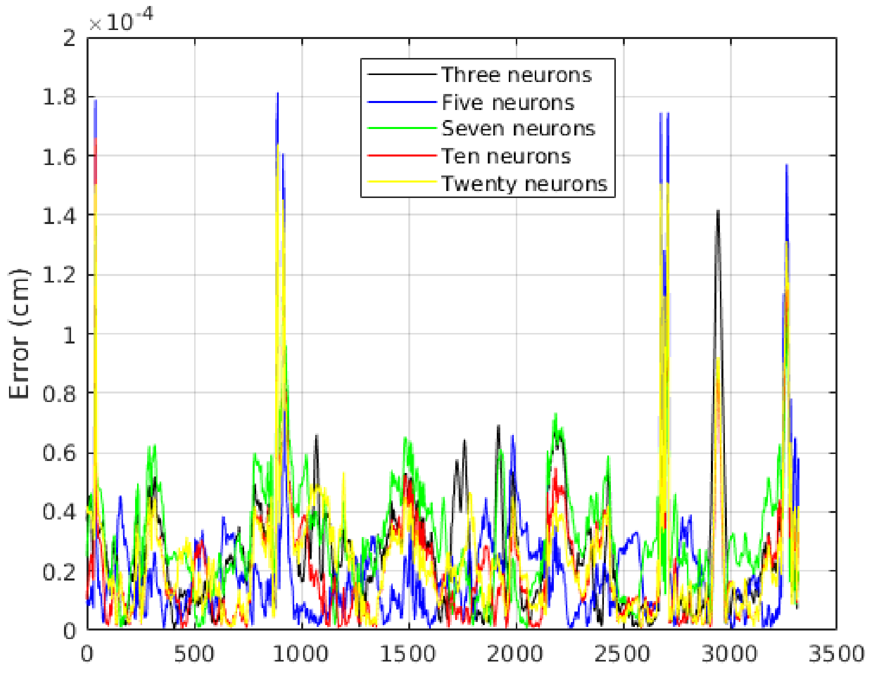


Figure 6: Error in cm, for architecture with two hidden layers.

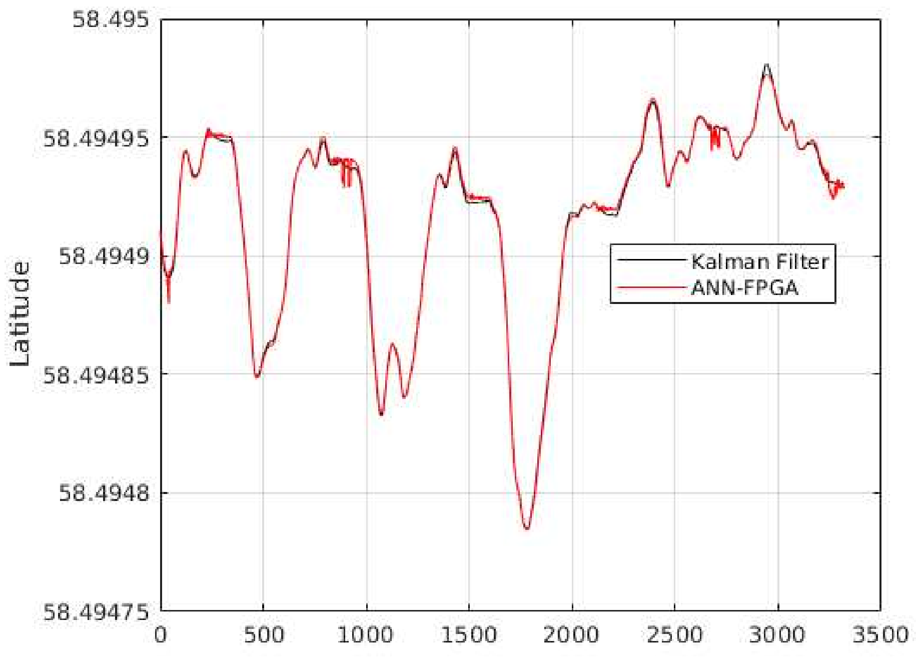


Figure 7: Latitude estimation of by ANN implemented in FPGA.

show a comparison between the result of the data fusion made by KF and the result of the fusion of data made by the neural fuser implemented in FPGA. the estimates for latitude and longitude – here the validation set was also used.

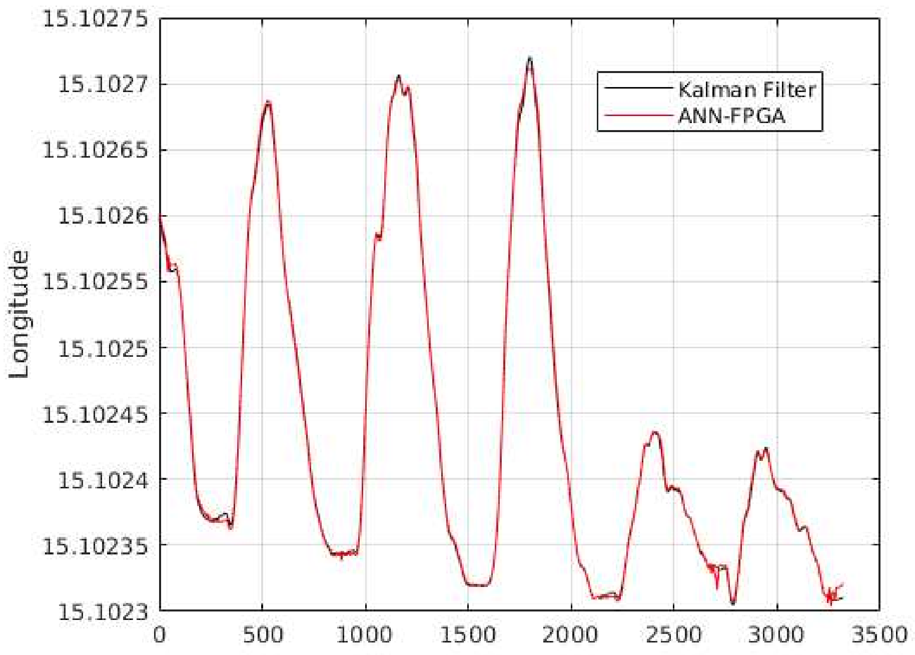


Figure 8: Longitude estimation of by ANN implemented in FPGA.

REFERENCES

Anochi J. and Velho H.C. Optimization of feedforward neural network by multiple particle collision algorithm. *FOCI IEEE Symposium on Foundations of Computational Intelligence*, pages 128–134, 2014.

Anochi J., Velho H.C., Furtado H., and Pacheco E.L. Self-configuring two types of neural networks by mpca. *Journal of Mechanics Engineering and Automation*, 5, 2015.

Braga J.R.G., Campos Velho H.F., Conte G., Doherty, and Shiguemori E.H. An image matching system for autonomous uav navigation based on neural network. *ICARCV International Conference on Control, Automation, Robotics and Vision*, pages 1–6, 2016.

Braga J.R.G., Campos Velho H.F., and Shiguemori E.H. Estimation of uav position using lidar images for autonomous navigation over the ocean. *ICST International Conference on Sensing Technology*, pages 811–816, 2015.

Conte G. and Doherty P. Vision-based unmanned aerial vehicle navigation using geo-referenced information. *EURASIP Journal on Advances in Signal Processing*, 1:308–387, 2009.

Dash J.P., Watt M.S., Pearse G.D., Heaphy M., and Dungey H.S. Assessing very high resolution uav imagery for monitoring forest health during a simulated disease outbreak. *ISPRS Journal of Photogrammetry and Remote Sensing*, 131:1–14, 2017.

Eltner A., Baumgart P., Maas H., and Faust D. Multi-temporal uav data for automatic measurement of rill and interrill erosion on loess soil. *Earth Surface Processes and Landforms*, 40:741–755, 2014.

Fiori L., Doshi A., Martinez E., Orams M.B., and B. B.B. The use of unmanned aerial systems in marine mammal research. *Remote Sensing*, 9:543, 2017.

Goltz G.A.M., Shiguemori E.H., and Campos Velho H.F. Position estimation of uav by image processing with neural networks. *X Congresso Brasileiro de Inteligência Computacional (CBIC-2011)*, pages 9–17, 2011.

Goncalves J. and Renato H. Uav photogrammetry for topographic monitoring of coastal areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 104:101–111, 2015.

Groves P.D. Principles of gnss, inertial, and multisensor integrated navigation systems, 2nd edition [book review]. *IEEE Aerospace and Electronic Systems Magazine*, 30(2):26–27, 2015.

James M., S. R., d'Oleire Oltmanns S., and U. N. Optimising uav topographic surveys processed with structure-from-motion: Ground control quality, quantity and bundle adjustment. *Geomorphology*, 280:51–66, 2017.

Valavanis K.P. and Vachtsevanos G.J. *Handbook of Unmanned Aerial Vehicles*, volume 1. Springer Publishing Company, Incorporated, 2014.