Pulse-coupled neural network performance for real-time identification of vegetation during forced landing

D. J. Warne[1] R. F. Hayward[2] N. A. Kelson[3]
J. E. Banks[4] L. Mejias[5]

(Received 19 December 2013; revised 5 March 2014)

Abstract

Safety concerns in the operation of autonomous aerial systems require safe-landing protocols be followed during situations where the mission should be aborted due to mechanical or other failure. This article presents a pulse-coupled neural network (PCNN) to assist in the vegetation classification in a vision-based landing site detection system for an unmanned aircraft. We propose a heterogeneous computing architecture and an OpenCL implementation of a PCNN feature generator. Its performance is compared across OpenCL kernels designed for CPU, GPU, and FPGA platforms. This comparison examines the compute

http://journal.austms.org.au/ojs/index.php/ANZIAMJ/article/view/7851 gives this article, © Austral. Mathematical Soc. 2014. Published March 21, 2014, as part of the Proceedings of the 11th Biennial Engineering Mathematics and Applications Conference. ISSN 1446-8735. (Print two pages per sheet of paper.) Copies of this article must not be made otherwise available on the internet; instead link directly to this URL for this article.

times required for network convergence under a variety of images to determine the plausibility for real-time feature detection.

Contents

1 Introduction

Critical to the integration of unmanned aircraft (UA) in civilian airspace is the capability to resolve on-board failures that can lead to an emergency landing (or forced landing) scenario. A UA forced landing can occur as a consequence of an unexpected event such as engine failure, hardware or software malfunction, or poor weather conditions [10]. The wide adoption of UAs in civilian settings will be conditional on their ability to demonstrate

equivalent levels of safety to that of manned aircraft [13, 3]. Hence, the ability for a UA to handle autonomously a forced landing is one of the most important technology enablers for their commercial use [10].

The goal of an autonomous UA forced landing system is to guide the aircraft to a suitable landing site with minimal functioning systems in the safest possible manner. A successful forced landing should minimise personal injury, minimise damage to property, and avoid damage to the UA itself. Three major components to this problem involve the selection of a suitable landing site, the decision making process, and the guidance of the aircraft to the landing site using minimal controls [10, 12, 9].

The detection, path planning, guidance and control aspects of this problem were investigated by Mejais et al. [10, 12, 11]. However, finding a computationally feasible solution to the landing site selection problem is still very much an unsolved problem [9]. Therefore, in this article we focus on the computational aspects associated with the landing site selection problem using a pulse-coupled neural network (PCNN) model.

A suitable landing site should be an area with smooth terrain free of buildings, infrastructure and dense vegetation. In particular, the identification of vegetation density is one of the most challenging aspects of the landing site selection problem. In this regard, biologically inspired PCNNs were shown to be particularly effective when applied to vegetation classification. Li et al. [8] applied a model based on PCNNs to classify tree species for the purpose of maintaining power lines. Our goal is to apply a similar model for texture-based classification of vegetation density to the aerial imagery collected from the UAs on-board cameras.

This article addresses the computational aspects of the PCNN model through the development of an Open Computing Language (OpenCL) implementation which are benchmarked on CPUs, GPUs, and FPGAs. Our benchmark tests show that an FPGA implementation achieves high performance and power efficiency for a low power overhead. As a result, real-time texture classification for a UA embedded environment is feasible using PCNN. The results of this

study also have wider applicability to PCNN-based algorithms on embedded systems in general.

2 Background

2.1 Pulse-coupled neural networks

Pulse-coupled neural networks are neural models for image processing which were biologically inspired by the structure of the visual cortex [4]. A PCNN consists of a two dimensional array of integrate-and-fire neurons where each neuron (i, j) receives stimuli from the (i, j)th pixel of an input image I, and from its neighbouring neurons $N(i, j)$. The pulsing neuron firing patterns are applied to feature extraction and classification of images [14].

The model used by Li et al. [8] is based on the unit-linking PCNN as defined by Gu et al. [7, 6]. Given an image I with K components, a unit-linking PCNN is defined by

$$F^t_{i,j} = \sum_{k=0}^{K} w_k I^t_{i,j,k}\,, \tag{1}$$

$$L^t_{i,j} = \begin{cases} 1 & \text{if } \left(\sum_{(x,y)\in N(i,j)} Y^{t-1}_{x,y}\right) > 0\,, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

$$U^t_{i,j} = \left(1 + \beta L^t_{i,j}\right) F^t_{i,j}\,, \tag{3}$$

$$Y^t_{i,j} = \begin{cases} 1 & \text{if } U^t_{i,j} > \theta^t_{i,j}\,, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

$$\theta^t_{i,j} = \theta^{t-1}_{i,j} - \alpha + VY^{t-1}_{i,j}\,, \tag{5}$$

where t is time, w_k is the weighting on the kth component of I, β is the weight on the link input, α is the threshold attenuation, and V is the threshold scale.

In equation (1) the pixel component stimuli are aggregated into a single neuron input feed which is modulated by the activity in the neurons neighbourhood in equations (2) and (3). The neuron fires if the modulated input exceeds the neurons threshold (equation (4)). If the neuron fires, then the threshold is increased in order to put the neuron into a refractory period (equation (5)), otherwise the threshold decreases linearly by α.

Li et al. [8] designed a scale and rotational invariant vegetation classifier using the pulse spectral frequency,

$$\begin{aligned} \mathrm{PSF}^t &= \frac{N^t}{N_{max}}, \\ N^t &= \sum_{i,j} Y^t_{i,j}, \\ N_{max} &= \max\left(N^0, N^1, \ldots, N^T\right), \end{aligned}$$

where T is the final iteration. It is expected that this classifier will significantly assist in the selection of a landing site during a UA forced landing.

2.2 Heterogeneous computing and OpenCL

Despite the simplicity of the unit-linking PCNN model, it is still too computationally expensive to execute in a reasonable time frame on a low-powered embedded processor as is required for a UA forced landing. Heterogeneous computing (i.e., multiple processor architectures working together to solve a single problem) could provide us with a feasible low power and high performance solution. To investigate a heterogeneous solution, our development was performed using OpenCL [5].

OpenCL is an open standard maintained by the Khronos Group for the programming of heterogeneous computing systems [5]. OpenCL consists of a parallel programming device language, and a host application programming interface (API) for communication with the parallel device. While the most

common implementations of OpenCL are for CPUs and GPUs (graphics processing units), the standard is designed to be extended to other specialised processors like digital signal processors (DSPs) and field-programmable gate arrays (FPGAs).

FPGAs are reconfigurable integrated circuits, consisting of programmable look-up tables (LUT), block RAM (BRAM), and programmable interconnects. An FPGA may be configured to implement application specific accelerators at runtime. Since FPGAs run at very low frequencies (e.g., 200–400 MHz), they are also low power and ideal for embedded applications.

Traditionally, FPGA development was a tedious digital circuit design process. A custom processor needed to be designed clock-by-clock at the register transfer level using a hardware design language. However, recently one of the major FPGA vendors, Altera, released an OpenCL software development kit (SDK) for their Stratix V series FPGAs [1]. This provides an environment for portable development across CPUs, GPUs, and FPGAs.

3 PCNN implementation

In this section we present an OpenCL implementation of the unit-linking pulse-coupled neural network model. The digital circuit that results from the OpenCL code will also be discussed. The target hardware is the Altera Stratix V FPGA packaged with the Bittware S5-HQ half-length PCIe board [2].

3.1 OpenCL programming model

OpenCL models a computing platform as a host sequential processor connected to one or more parallel compute devices. Each compute device typically consists of a number of compute units, which are further broken down into simple scalar processing elements, as shown in Figure 1. The exact mapping

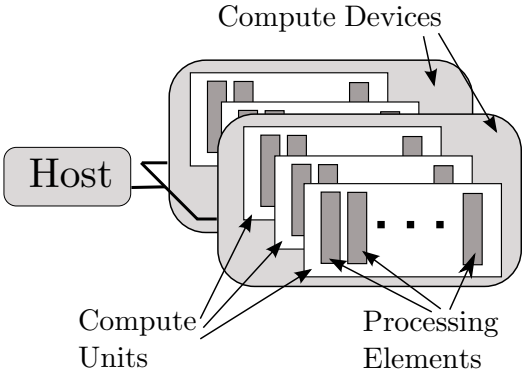

Figure 1: The OpenCL view of a computing platform.

of compute units and processing elements depends on the physical device hardware; for example, a compute unit could be a single CPU core, or a GPU streaming multi-processor. A processing element is a simple subunit of the compute unit, such as a single element of a CPU SIMD unit.

In the more usual case of CPU or GPU compute devices, OpenCL device code will compile to micro code for the appropriate device, which will execute on the available physical compute units and processing elements. This should be contrasted with the FPGA case where the physical mapping of compute units and processing elements is generated by the OpenCL device code itself. The OpenCL device code 'compiles' to a configuration image to implement the required physical compute units and processing elements in reconfigurable hardware. In other words, OpenCL device code for a FPGA does not define executable code, but rather defines a co-processor architecture which implements the algorithmic behaviour of the code natively.

3.2 The PCNN processor architecture

The executable object in an OpenCL device program is a compute kernel. Multiple kernel instances (called work items) will execute on processing

Algorithm 1: $Y = \text{PCNN\_KERNEL}(F)$: PCNN compute kernel logic

{F *is the input feed in global memory*}
{*Define two local caches to reduce global memory I/O.*}
$\text{CACHE}_{\text{in}}[16][16]$
$\text{CACHE}_{\text{out}}[16][16]$
$[i_G, j_G] \leftarrow \text{GET\_GLOBAL\_INDEX}$ {*Index relative to image.*}
$[i_L, j_L] \leftarrow \text{GET\_LOCAL\_INDEX}$ {*Index relative to work group.*}
$\text{CACHE}_{\text{in}}[i_L][j_L] \leftarrow F[i_G][j_G]$
SYNCHRONISE {*Wait until this work group of neurons is loaded.*}
$\text{CACHE}_{\text{out}}[i_L][j_L] \leftarrow \text{NEURON}(\text{CACHE}_{\text{in}}[i_L - 1 : i_L + 1][j_L - 1 : j_L + 1])$
SYNCHRONISE {*Wait until this work group of neurons is processed.*}
$Y[i_G][j_G] \leftarrow \text{CACHE}_{\text{out}}[i_L][j_L]$
return

elements. Work items form work groups; these will execute on the same compute unit and have access to shared local memory resources. In the FPGA case, compute units and processing elements are generated which exactly implement the behaviour of the work items and work groups.

The PCNN OpenCL kernel code implements the behaviour of a single work item within a work group. The work item of the PCNN is a single neuron, and this generates a single neuron processing element. Neuron work items are grouped into 16×16 work groups to minimise global memory reads. The pseudo-code for a single neuron is shown in Algorithm 1.

The resulting compute unit generated has 16×16 neuron processing elements and a local memory cache capable of storing an 18×18 section of the current neuron firing state; this cache is constructed using the FPGA BRAM. Up to six of these compute units can be implemented concurrently using the available hardware resources on the Stratix V. Clocked at approximately 160 MHz, this processor has a theoretical peak performance of around 700 million neurons per second. A diagram of the PCNN processor architecture is provided in Figure 2.

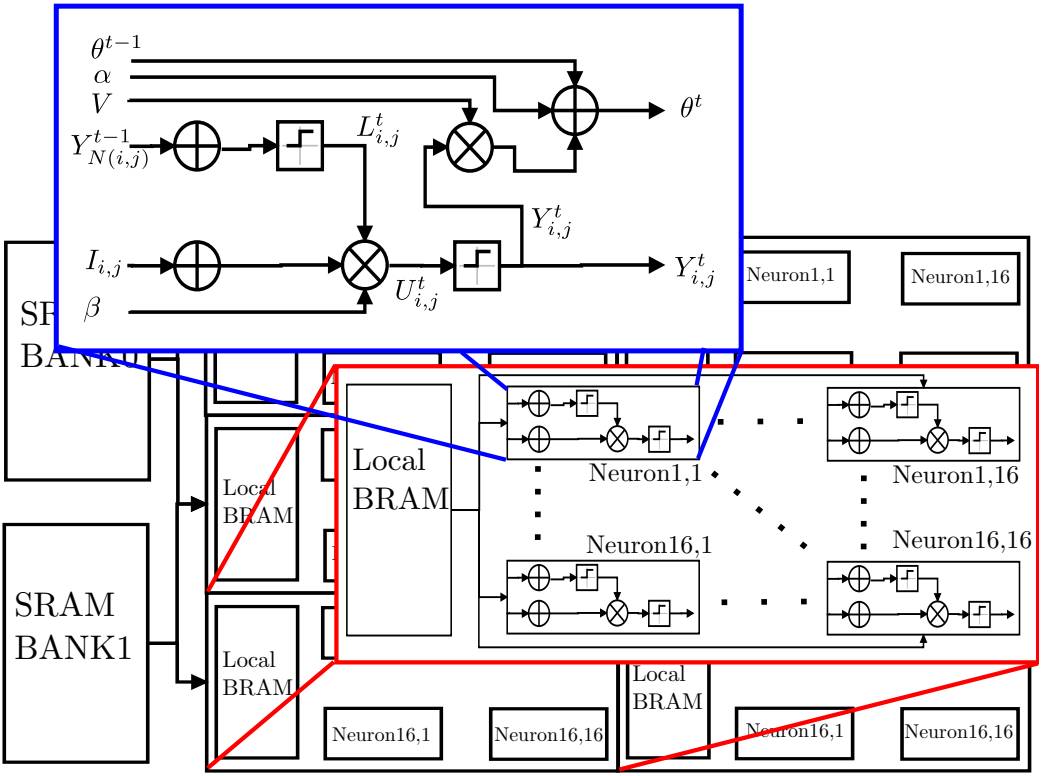

Figure 2: Overall PCNN processor architecture.

4 Performance

Due to the portability of OpenCL, a performance comparison across multiple architectures is done with relatively minor code modifications, such as the addition of vendor specific kernel attributes to guide circuit synthesis. We compared the performance of the PCNN model executing on a single core of an Intel E5-2670 CPU (clocked at 2.6 GHz), an NVidia Quadro 4000 GPU (clocked at 950 MHz), and the Altera Stratix V FPGA (clocked at 168 MHz). All host and device code was compiled using default compiler options. This section

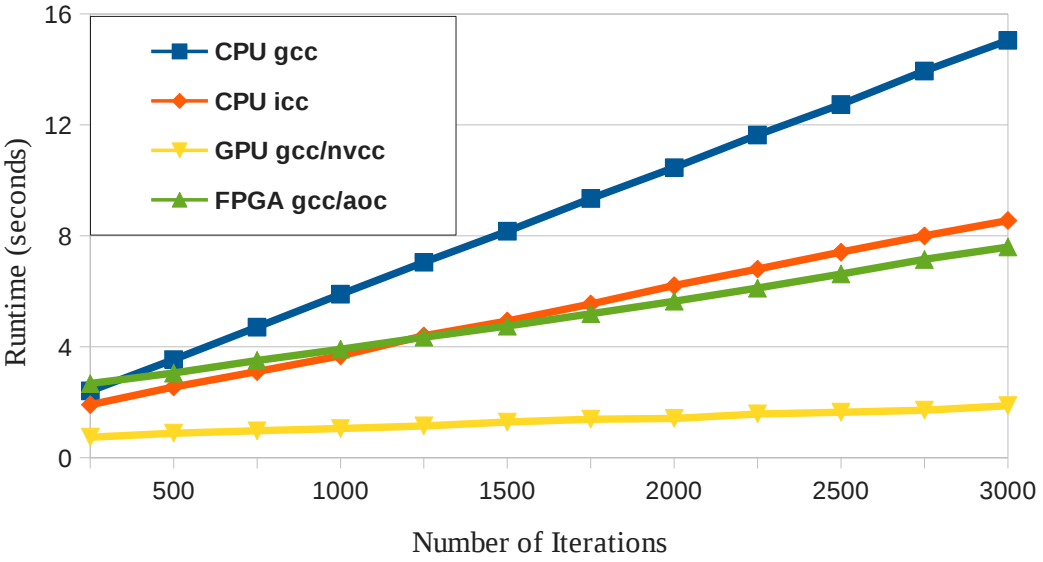

Figure 3: PCNN performance comparison for CPU (Intel E5-2670), GPU (NVidia Quadro 4000), and FPGA (Altera Stratix V)

presents the results of our comparison in terms of raw compute performance, power consumption and efficiency.

4.1 Compute performance

The comparison benchmark consisted of executing the PCNN model over the well known Brodatz texture database[1]. The number of iterations was varied from $T = 250$ to $T = 3000$. Figure 3 shows the benchmark results.

In terms of runtime, the GPU was the best performing with a speedup of approximately four times over the CPU code compiled with the Intel

[1] http://multibandtexture.recherche.usherbrooke.ca/original_brodatz.html

compiler[2](i.e., `icc`) running on a single core. The FPGA solution performed slightly better than the Intel compiled CPU version. This result is very impressive for the FPGA, especially when the processor clock speeds are taken into acount. The CPU is clocked at 2.6 GHz whereas the FPGA is only clocked at 168 MHz; this means the FPGA performs roughly 16 times more work per clock cycle.

4.2 Power efficiency and consumption

In the context of a UA embedded system, power consumption and efficiency is of the utmost importance [9]. Power efficiency is the amount of compute performed for the amount of extra power consumed by the system, and power consumption is the total power consumed under load. Figure 4 shows measured power consumption at idle and under load, and Figure 5 shows the efficiency of the compute benchmark results from Section 4.1. A standard wall socket power meter (A Watts Clever EW-AUS5001, accuracy ±0.5%) was used to take power measurements.

The FPGA solution has comparible power efficiency to the GPU solution and this FPGA efficiency is achieved with a significantly lower total power consumption than both GPU and CPU solutions. With such a low total power footprint, the results for the FPGA based PCNN show promise for vision-based landing site detection on a UA embedded system.

5 Conclusion

By implementing the PCNN model in OpenCL, we were able to target a range of different processor architectures. The FPGA based OpenCL PCNN

[2]The Intel compiler CPU code attains a speedup of approximately 2× over the GNU compiled version (i.e., `gcc`). Using vendor specific compilers is important for a fair comparison.

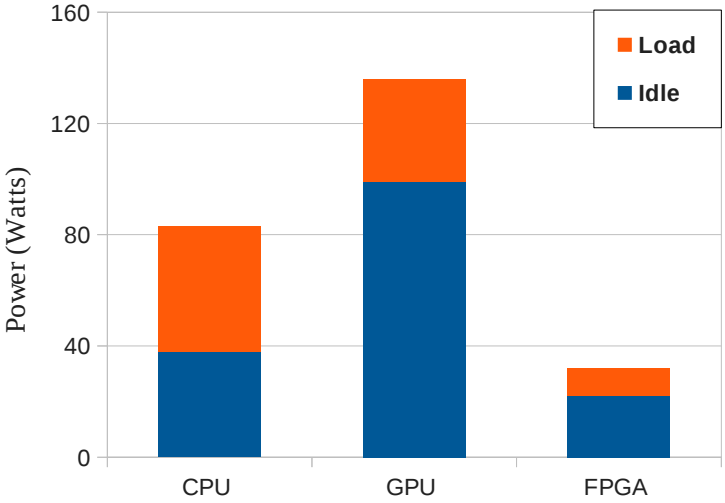

Figure 4: Power consumption comparison.

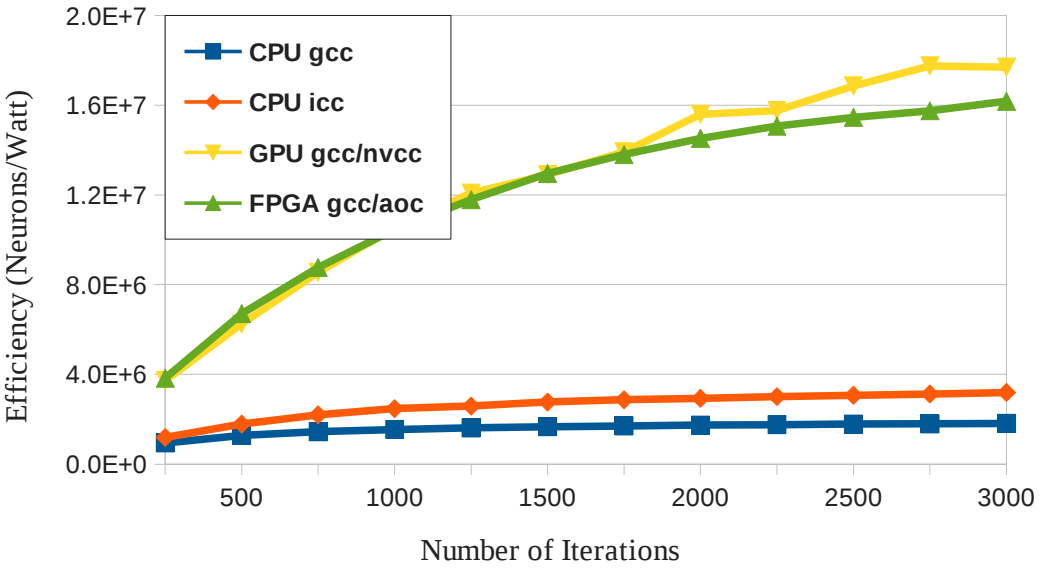

Figure 5: Power efficiency comparison.

implementation achieves the raw performance of a high performing server CPU, the power efficiency of a workstation GPU, and a total power consumption significantly lower than both CPU and GPU based PCNN models. The results indicate the feasibility of using an FPGA based PCNN model for real-time vegetation classification for the purposes of landing site identification during a UA forced landing. Furthermore, the feasibilty of PCNN solutions in embedded systems in general is greatly enhanced by this work.

Acknowledgements This research forms part of Project ResQu led by the Australian Research Centre for Aerospace Automation (ARCAA). The authors gratefully acknowledge the support of ARCAA and the project partners, Queensland University of Technology (QUT); Commonwealth Scientific and Industrial Research Organisation (CSIRO); Queensland State Government Department of Science, Information Technology, Innovation and the Arts; Boeing and Insitu Pacific.

References

[1] Altera. Implementing FPGA design with the OpenCL standard. White Paper, Altera Inc., November 2012. http://www.altera.com/literature/wp/wp-01173-opencl.pdf C6

[2] Bittware. S5-PCIe-HQ user guide. Technical report, Bittware, Inc., September 2013. C6

[3] M. T. DeGarmo. Issues concerning integration of unmanned aerial vehicles in civil airspace. Technical Report mP 04W0000323, MITRE Corporation, 2004. https://www.mitre.org/sites/default/files/pdf/04_1232.pdf C3

[4] R. Eckhorn, H. J. Reiboeck, M. Arndt, and P. W. Dicke. A neural network for feature linking via synchronous activity: Results from a cat visual cortex and from simulations. In *Models of Brain Function*, pages 255–272, Cambridge, UK, 1989. Cambridge University Press. C4

[5] Khronos OpenCL Working Group. The OpenCL specification. Technical report, Khronos Group, October 2009. http://www.khronos.org/registry/cl/specs/opencl-1.0.pdf C5

[6] X. Gu, Y. Fang, and Y. Wang. Attention selection using global topological properties based on pulse coupled neural network. *Computer Vision and Image Understanding*, 117:1400–1411, 2013. doi:10.1016/j.cviu.2013.05.004 C4

[7] X. Gu, L. Zhang, and D. Yu. General design approach to unit-linking pcnn for image processing. In *Proceedings of International Joint Conference on Neural Networks*, pages 1837–1841, 2005. doi:10.1109/IJCNN.2005.1556159 C4

[8] Z. Li, R. F. Hayward, R. A. Walker, and Y. Liu. A biologically inspired object spectral-texture descriptor and its application to vegetation classification in power-line corridors. *IEEE Geoscience and Remote Sensing Letters*, 8(4):631–635, 2011. doi:10.1109/LGRS.2010.2098391 C3, C4, C5

[9] A. Lu, W. Ding, J. Wang, and H. Li. Automonmous vision-based safe area selection algorithm for UAV emergency forced landing. In *Proceedings of the International Conference on Information and Computer Applications 2012*, pages 254–261, 2012. doi:10.1007/978-3-642-34041-3_37 C3, C11

[10] L. Mejias and P. Eng. Controlled emergency landing of an unpowered unmanned aerial system. *Journal of Intelligent and Robotic Systems*, 70:421–435, 2013. doi:10.1007/s10846-012-9767-5 C2, C3

[11] L. Mejias and D. L. Fitzgerald. A multi-layered approach for site detection in uas emergency landing scenarios using geometry-based image segmentation. In *Proceedings of the 2013 International Conference on Unmanned Aerial Systems*, pages 366–372, Atlanta, Georgia, 2013. IEEE Control Society. doi:10.1109/ICUAS.2013.6564710 C3

[12] L. Mejias, D. L. Fitzgerald, P. C. Eng, and L. Xi. Forced landing technologies for unmanned aerial vehicles : towards safer operations. In *Aerial Vehicles*, pages 415–442, Kirchengasse, Austria, 2009. In-Tech. doi:10.5772/6481 C3

[13] US-OSoD. Unmanned systems integrated roadmap fy2011-2036. Technical report, Office of the Secretary of Defense, US, 2011. http://info.publicintelligence.net/DoD-UAS-2011-2036.pdf C3

[14] Z. Wang, Y. Ma, F. Cheng, and L. Yang. Review of pulse-coupled neural networks. *Image and Vision Computing*, 28:5–13, 2010. doi:10.1016/j.imavis.2009.06.007 C4

Author addresses

1. **D. J. Warne**, High Performance Computing and Research Support, Queensland University of Technology, Queensland 4001, Australia.
mailto:david.warne@qut.edu.au

2. **R. F. Hayward**, School of Electrical Engineering and Computer Science, Queensland University of Technology, Queensland 4001, Australia.
mailto:r.hayward@qut.edu.au

3. **N. A. Kelson**, High Performance Computing and Research Support, Queensland University of Technology, Queensland 4001, Australia.
mailto:n.kelson@qut.edu.au

4. **J. E. Banks**, School of Electrical Engineering and Computer Science, Queensland University of Technology, Queensland 4001, Australia. mailto:j.banks@qut.edu.au

5. **L. Mejias**, Australian Research Center for Aerospace Automation, Queensland University of Technology, Queensland 4001, Australia. mailto:luis.mejias@qut.edu.au