FPGA Implementation of an Evolutionary Algorithm for Autonomous Unmanned Aerial Vehicle On-Board Path Planning

Jonathan Kok, Felipe Gonzalez, and Neil Kelson

***Abstract*—In this paper, a hardware-based path planning architecture for unmanned aerial vehicle (UAV) adaptation is proposed. The architecture aims to provide UAVs with higher autonomy using an application specific evolutionary algorithm (EA) implemented entirely on a field programmable gate array (FPGA) chip. The physical attributes of an FPGA chip, being compact in size and low in power consumption, compliments it to be an ideal platform for UAV applications. The design, which is implemented entirely in hardware, consists of EA modules, population storage resources, and three-dimensional terrain information necessary to the path planning process, subject to constraints accounted for separately via UAV, environment and mission profiles. The architecture has been successfully synthesised for a target Xilinx Virtex-4 FPGA platform with 32% logic slices utilisation. Results obtained from case studies for a small UAV helicopter with environment derived from LIDAR (Light Detection and Ranging) data verify the effectiveness of the proposed FPGA-based path planner, and demonstrate convergence at rates above the typical 10 Hz update frequency of an autopilot system.**

***Index Terms*—Evolutionary algorithm, field programmable gate array, path planning, unmanned aerial vehicle.**

I. Introduction

UNMANNED aerial vehicles (UAVs) are on an upsurge to be the preeminent platform for some military [2] and civilian aerial applications [3]. In the 5 years, 2010-2015, the U.S. UAV market alone is forecasted to generate USD$62 billion in revenues [4]. Potential advantages of UAV deployment on both civilian and military missions include decreased risk of fatality and reduced workload of human operators, these in turn increasing efficiency and reducing costs of missions.

The advancement of UAV technology and applications has been assisted by research aimed at improving levels of autonomy whilst being bounded by flight constraints. These include power consumption, physical space and weight limitations, along with limited on-board telecommunication, computational, and other resources. One aspect of the research effort is to investigate the feasibility of more autonomous on-board path planning as an alternative to UAV path planning currently performed remotely by human operators. This is not straightforward, as solving the path planning problem for autonomous UAV navigation is an NP-hard problem [5]. Furthermore, the overall path planning process involves complex design issues, as there exists strong couplings between the environment representation, type of path planning algorithm and the application-specific task [6].

To date, UAV path planning algorithms have been developed via mainly PC software-based implementation, which may not explicitly take into account some required real-time flight constraints appropriate for UAVs operating autonomously in the field [7]. In contrast, reconfigurable field programmable gate arrays (FPGAs) are relatively small and light, making them suitable for flight-constrained UAV applications where e.g. the size and weight of the payload is greatly restricted.

In light of the above, our work considers an entirely on-board hardware-based path planning system in the context of an overall constrained process, where profiles that define constraints relating to the environment, the mission, and the individual characteristics of the UAV are specified. A path planning architecture is proposed and implemented directly on an FPGA platform based on the use of evolutionary algorithms (EAs). Regarding algorithm choice, EAs are considered as viable search algorithms for real-time UAV path planning [8]–[13].

The EA-based path planner developed in this work is based on a modified genetic algorithm (GA) [14] capable of finding global solutions, albeit at significant computational expense [15]. To address this issue, a GA-based path planner is developed with all modules of the proposed architecture being entirely implemented and run on a single FPGA for computational efficiency. Compared to prior work, this study is the only one known to the authors that attempts to implement all of the relevant functionalities of the GA-based path planner entirely on reconfigurable FPGA hardware. Conceptually, we elaborate on earlier design proposals by explicitly accounting for a wider range of flight constraints via the inclusion of segregated UAV, environment and mission profiles within corresponding separate hardware modules. Additionally, three-dimensional terrain data is stored on the FPGA, via the hardware module corresponding to the environment profile. The approach taken is intended to be more modular and extensible than prior proposals, and should permit greater UAV autonomy.

To the Authors knowledge, optimal in-hardware implemen-

J. Kok and F. Gonzalez are with the Australian Research Centre for Aerospace Automation (ARCAA), Eagle Farm, Queensland 4009 Australia (e-mail: jonathan.kok@student.qut.edu.au; felipe.gonzalez@qut.edu.au).

N. Kelson is with the High Performance Computing and Research Support Group, Division of TILS, Queensland University of Technology, Brisbane 4001 Australia. Brisbane, Queensland 4000 Australia (e-mail: n.kelson@qut.edu.au).

[1]This paper is an extension of earlier work which appeared in [1] J. Kok, F. Gonzalez, N. Kelson, T. Gurnett, and R. Walker, "A synthesizable hardware evolutionary algorithm design for unmanned aerial system real-time path planning," in *Proceedings of the 2010 Australasian Conference on Robotics & Automation*, 2010.

tation of various GA functions and modules has also not been addressed in detail by earlier studies. In contrast, we instead explore hardware-based implementation of both the overall architecture and the various functional modules within it so as to exploit the parallel processing capability of the FPGA. For example, our design incorporates a 6-to-1 multiplexer memory interface which is resource efficient as compared to passing the entire GA population from module to module. As shown via three case studies, a hardware-based implementation of the proposed architecture which exploits the processing capability of the FPGA can achieve convergence at rates above the typical 10 Hz update frequency of an autopilot system.

This paper is organized as follows. Section II gives further details of related work and highlights the main difference in our approach. Section III presents our proposed FPGA-based UAV path planning system, including descriptions of the architecture, system operation, execution flow and communication. Section IV provides hardware implementation details for a target Xilinx Virtex 4 FPGA development platform (available from the university HPC department). Choices for GA population characteristics and other parameters appropriate to the available resources on the target FPGA are presented, along with details of in-hardware implementations of various path planner modules. Section V presents results of empirical case studies to verify the effectiveness of the proposed FPGA-based path planner for a specific UAV over a sample three-dimensional terrain. The chosen UAV is an unmanned 1 m length helicopter with a limited 5 kg payload capacity (available from the ARCAA), while the terrain is a 512 m^3 environment derived from LIDAR (Light Detection and Ranging) data. Section VI concludes with a brief summary of the applicability and current research direction.

II. Background and Related Work

A. UAV Path Planning

Path planning can be defined as the framework employed to determine flight plans for UAVs traversing from one location to another [7].

B. Previous FPGA Implementations for Path Planning

FPGAs are and have been used for a range of engineering applications [16], however their influence in path planning applications is recent and limited [15], [17]–[23].

Alliare *et al.* [15] demonstrated that the possibility of increased UAV navigation autonomy can be achieved by instantaneous on-the-fly replanning via the implementation of a path planning GA on an FPGA for algorithm acceleration. They argue that GAs produce higher quality solutions as compared to deterministic algorithms but are disadvantaged due to their extensive computational overhead that is inevitably inherited by the GA's population-based metaheuristics optimisation approach. Their path planning GA implementation details were partially set according to Cocaud's [14] work involving a customised GA specifically for UAV task allocation and path generation. Their results indicate that some mechanisms of the GA running on an FPGA can be sped up by factor of thousands. However, their research was limited to co-simulation between a CPU and an FPGA running simultaneously and exchanging information in a collaborative manner.

Vacariu *et al.* [17] proposed an FPGA implementation of a simple breadth-first search (BFS) algorithm [24] applied on static two-dimensional discrete environment. The BFS technique, which has two degrees of freedoms, is based on maintaining a queue of all accessible neighbours, whereby a sequence of directions leading to the goal point is acquired. This search algorithm is complete, that is to say it will find a solution if one exists, but does not guarantee any level of optimality or feasibility. Their results show execution times sped up by factors of hundreds.

Girau *et al.* used an FPGA to compute approximated harmonic control functions [25] for making robot navigation decisions. The use of harmonic functions ensures that generated trajectories avoid local optima in cluttered and concaved environments [26]. They argued that the main advantage of their work was not the speedup, as real-time computational speed can be easily reached by software coded harmonic control functions. Instead, they highlighted the potential of embedding FPGAs for online processing needs on low powered mobile robots.

Sudha and Mohan [19] designed an FPGA-accelerated path planner based on the Euclidean distance transform [27] of a captured image from an overhead camera. Priya *et al.* [20] customised an FPGA architecture for path planning based on revised simplex method [28] applied on a pre-constructed visibility graph [29]. Sudha and Mohan argue that their path planning solution is process complete as contrasted to Priya *et al.* work, as a raw binary image of environment is directly processed in the hardware rather than a meta-modelled nodes-and-edges visibility graph. On the other hand, results from Priya *et al.* were in factors of μs as compare to ms from Sudha and Mohan: the former was interested in ballistic missiles application, whereas the latter was directed towards ground-based robot navigation.

Hachour [21] proposed an FPGA-based path planning GA method for ground-based mobile robots. However, the path planning GA concept and the actual hardware implementation were not described in any detail, and the results and validated functionality were not reported.

Huang *et al.* [22] proposed a hardware/software co-designed parallel elite genetic algorithm (PEGA) for ground mobile robot path planning in a static environment. Their FPGA-based PEGA architecture consists of two path planning GA [30] of which the evolutionary-influenced selection, crossover, and mutation modules operate concurrently. Elitism is preserved via a migration module that periodically exchanges the corresponding two best members into the selection pool after a predefined number of generations. However, the computationally dominant fitness evaluation function was executed sequentially on an embedded processor.

Schmidt and Fey [23] implemented marching pixels (MP) [31] applied on a skeleton map (SM) [32] on an FPGA for path planning. MP is akin to artificial ants behaving as modelled by cellular automata, where one pixel of an image is represented by a two-dimensional coordinate on the map. Their results show computational effectiveness in factors of ms for image

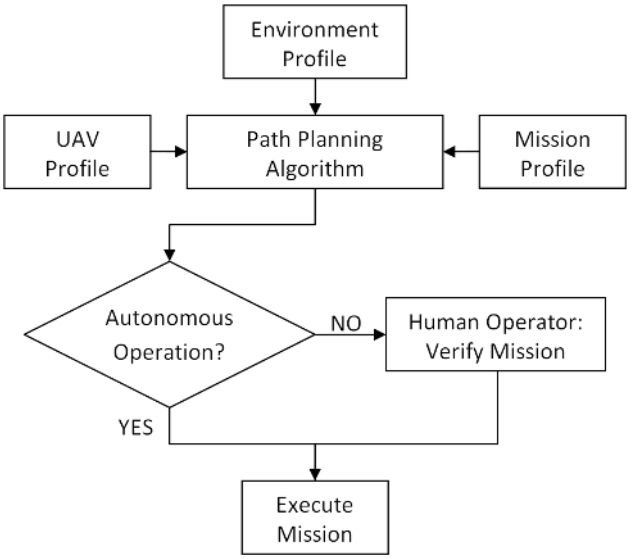

Fig. 1. Constraints associated with a typical UAV path planning process.

resolutions up to 1024×800. However, their approach inherits the two main disadvantages from the nature of SM sampling: the resulting path solution is non-optimal, and sharp turning edges are probable.

While the above research works summarised in Table I have established the concept of FPGA implementations for speeding up different computationally intensive path planning algorithms, this work instead aims to contribute a completely embedded UAV path planning system inclusive of constraints via UAV, environment and mission profiles. Note that for this work, the UAV was the designated robotic platform but this is not expected to be restrictive. Furthermore, unlike prior works which were limited to a two-dimensional model, except for [15], our work considers the real-world aspects of a three-dimensional model.

III. Proposed FPGA-based Path Planning System

A. Path Planning Constraints and Execution

As previously mentioned, here we consider path planning in the context of an overall constrained process. A diagram illustrating our approach is shown in Fig. 1. Flight constraints are segregated and handled separately with the aim of making the overall path planning architecture more easily generalisable and adaptable for different UAV, environment or mission configurations.

The environment profile proposed here contains terrain specific information by which the path planning algorithm is constrained. Note that no restriction on the source of data contained therein is intended for this or the other profiles. Similarly, in-flight data refresh of one or more profiles is not excluded. For example, static data for FPGA upload could be transmitted from the base station, while dynamic data could be generated by on-board cameras, sensors, or auxiliary UAVs.

The UAV profile defines the aircraft performance constraints of the targeted UAV platform such as minimum and maximum velocity, maximum fuel capacity, and turning radius. Similarly,

```
Modified Genetic Algorithm ()

    Initialise Population
    Loop
        Selection
        Genetic Operation
        Evaluation
        Population Update
        If ( Premature Convergence )
           Reinitialise Parts of Population
        End-If
    End-Loop
```

Fig. 2. Pseudo code of modified genetic algorithm for the path planner.

the mission profile includes all desired mission characteristics such as starting and ending coordinates, flight boundaries, elevation deviation, and maximum allowed mission-time.

The figure also illustrates a basic difference between autonomous and non-autonomous UAV flights. In non-autonomous operation, a human operator would verify the feasibility of the generated path solution and approve its execution. The proposed FPGA-based architecture instead aims to provide real-time solutions of practical utility for increasing the level of autonomy and confidence of the system, hence reducing or even eliminating the need for intervention by a human operator. This is useful in tedious missions (such as sampling and inspection).

B. Architecture

As noted earlier, the architecture for the proposed FPGA-based path planning design is based on an application specific genetic algorithm (GA) characteristic recommended by Cocaud [14] for flight path planning. Briefly, the GA is a stochastic optimisation method that iteratively generates improving candidate solutions using adaption techniques inspired by natural evolution, such as natural selection, recombination/crossover, mutation and inheritance, fitness evaluation and so forth [35], [36]. Elitism is included to prevent the loss of beneficial solutions during the evolutionary cycle. Note also that here the usual GA iterative process has been modified to reinitialise parts of the population diversity, if necessary, to conteract any tendancy towards premature convergence. Pseudo code of the modified GA used here is shown in Fig. 2. The various evolutionary operations are customised specifically for a path planning task, details of which will be elaborated below.

A schematic of the proposed FPGA-based path planning architecture is illustrated in Fig. 3. The design is intended to fit into a single FPGA, and includes modules corresponding to typical GA tasks where the functionality of each module can be set according to the algorithmic requirements of the specific GA under consideration.

C. Overall System Operation

Overall, the driving component of the FPGA-based path planning system shown in Fig. 3 is the Control Unit (CU) that monitors the evolutionary process throughout the entire

TABLE I
EXISTING LITERATURE ON FPGA IMPLEMENTATIONS OF PATH PLANNING ALGORITHMS

Work	Path Planning Algorithm	Environment Sampling Method	Platform Model
[15]	Modified Genetic Algorithm [14]	Hybrid Octree Knowledge Base [14]	Xilinx Virtex-II Pro XC2VP30
[17]	Breath-First Search Algorithm [24]	Sukharev Grid [33]	Xilinx Virtex-II Pro
[18]	Harmonic Potential Trajectory Control [25]	Sukharev Grid [33]	Xilinx Virtex-II XC2V6000
[19]	Euclidean Distance Transform [27]	Euclidean Distance Mapping [34]	Xilinx Spartan-3 XC3S1500L
[20]	Revised Simplex Method [28]	Visibility Graph [29]	Xilinx Virtex-II XC2V6000
[21]	Motion-based Genetic Algorithm [21]	-	Xilinx XC4000
[22]	Modified Genetic Algorithm [30]	Sukharev Grid [33]	Altera Stratix EP1S10F780C6
[23]	Marching Pixel [31]	Skeleton Map [32]	Xilinx Virtex-5 XC5VLX110T
Proposed	Modified Genetic Algorithm	Sukharev Grid [33]	Xilinx Virtex-4 XC4VLX200

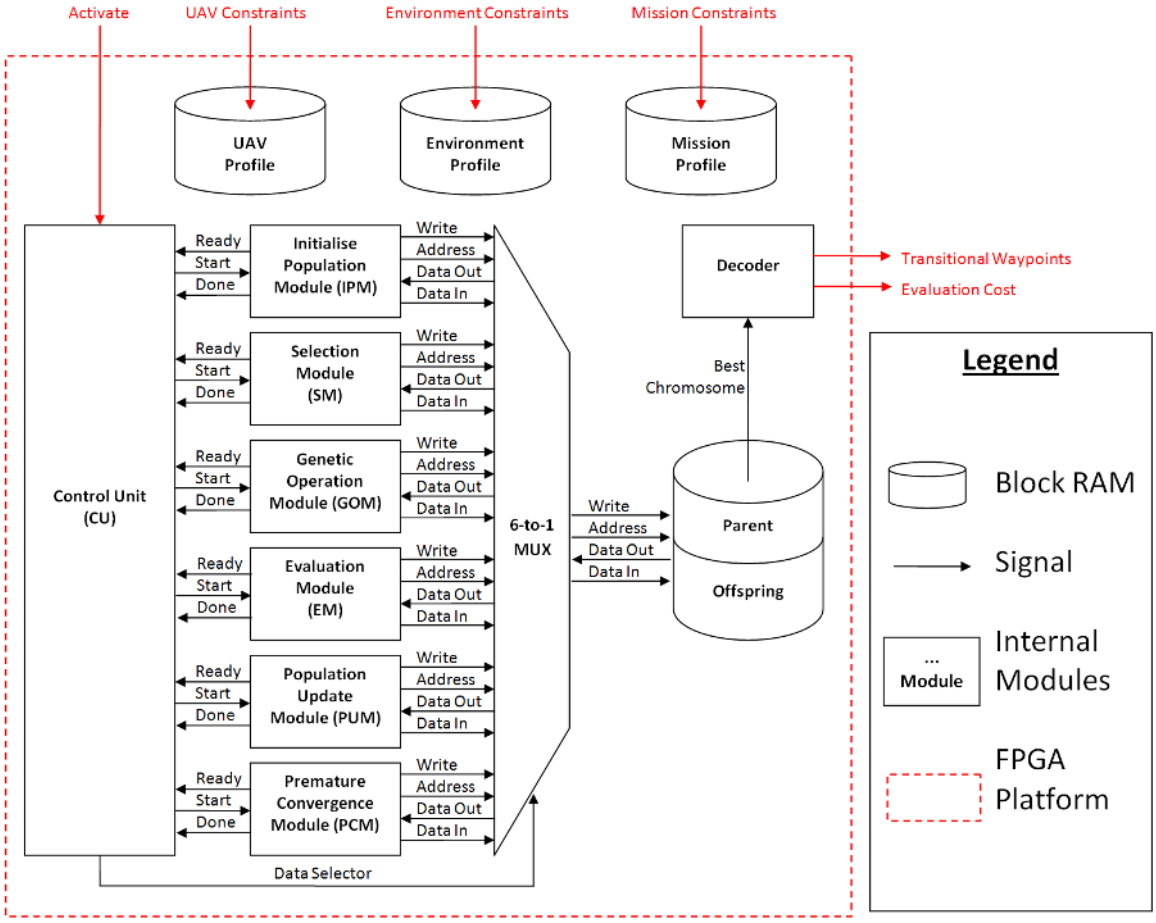

Fig. 3. Schematic of the FPGA-based path planner architecture.

operation. Random number generators modelled by cellular automata technique [37] are used for generating a sequence of pseudo random numbers. Block random-access memories (RAMs) in the FPGA are used for storing the parent and offspring populations, as well as the UAV, environment and mission profiles. The initial and subsequent populations are then evolved based on this information.

Population of chromosomes are stored in the Parent and Offspring memories. Each chromosome is made up of one possible path solution and its associated fitness (i.e. collision and distance costs for the UAV to traverse the given environment between designated start and end waypoints). An on-module memory interface is provided by the 6-to-1 multiplexer (MUX) for allowing all modules to read and write the Parent and Offspring memories.

The Decoder is used for converting the bitstream format of the best chromosome into a meaningful representation for the external output interface. Two inputs which drive the FPGA-based planner are the clock signal and the activate signal. The clock signal is connected to the embedded global clock of the FPGA, and the activate signal is connected to an input external to the FPGA. The best chromosome within the Parent memory

is connected to the output allowing a best path solution to be constantly available.

The Initial Population Module (IPM) is used for generating a population of candidate solutions based on the output from the random number generator. The Selection Module (SM) is used for applying selection pressure on the parent population to populate a mating pool of useful chromosomes. The Genetic Operation Module (GOM) is used for adaptively altering the chromosomes in the mating pool. The Evaluation Module (EM) is used for evaluating the altered chromosomes. The Population Update Module (PUM) is used for updating the next generation of parent population with useful chromosomes amongst the parent population and the newly generated offspring population. The Premature Convergence Module (PCM) is used for monitoring the gene diversity of the parent population, and signifies a possible premature convergence when diversity is lost early in the generational cycle.

D. Iterative Execution Flow and Communication

The flow of execution and communication between the individual units of the FPGA-based path planner is now described. Note that during the evolution phases, processes in each module are handled concurrently.

To begin the FPGA-based path planning process, an activate signal is received, initiating the IPM to generate, influenced by flight parameters such as minimum and maximum UAV elevation, a set of random path solutions that are evaluated internally and stored in the Parent memory. This completes the initial setup of the FPGA-based planner.

To commence the next iteration of the EA process, the IPM notifies the CU that the FPGA-based planner is ready to begin execution. The SM exercises a selection process on the Parent memory and populates the Offspring memory with a mating pool of elected chromosomes. The GOM, which consists of the genetic crossover and mutation operations, starts genetic operation on the Offspring memory, creating new chromosomes. Once completed, the genetically altered chromosomes are updated back into the Offspring memory. The EM evaluates the feasibility and fitness of new chromosomes generated by the GOM. Upon completion, the EM updates the fitness information within each chromosome and sends the evaluated chromosomes back into the Offspring memory. The PUM sorts and updates the Parent memory for the next generation with elites from amongst the Parent and Offspring memory. The PCM monitors the Parent memory for premature convergence, and triggers a reinitialise signal if necessary. Finally, the CU excludes the IPM in the next evolution cycles until the reinitialise signal is activated.

The above iterative steps are repeated endlessly. An independent Decoder transmits out the optimised flight path solution at every clock cycle, this being the best chromosome decided through the fitness sorting PUM. The GA run is never ceasing, hence it is always trying to improve its current best path. The system only restarts when the activate signal is triggered again. Such a scenario may occur, for example, when the UAV has reached the end waypoint or new environment or mission profiles are uploaded. Effectively, the planning of a longer journey across a dynamic environment would be carried out in a stepwise manner.

IV. Hardware Implementation Details

To explore the feasibility of the proposed design architecture, an implementation of the proposed FPGA-based path planner in synthesisable very-high-speed integrated circuit hardware description language (VHDL) was undertaken targeting an available development platform containing a Xilinx Virtex-4 LX200 (XC4VLX200-11FF1513) FPGA processor. This platform was used to explore implementation issues such as the population characteristics and extent of parallelism possible within the design, subject to various FPGA hardware-specific constraints including the programmable logic resources available on the device. The implementation of the FPGA-based planner requires application of the GA population characteristics and operations specifically for path planning. As shown in Fig. 3, selection, genetic operation, evaluation, and update operation are involved and require specification in the iterative GA process. Implementation details of the various modular elements and the GA population characteristics are briefly described below.

A. Implementation-Specific Design

1) Encoding of GA Parameters: One of the first design decisions is determining the encoding of the parameters as this will, for example, directly affect usage of available resources on the target Virtex 4 FPGA and also enhance or hinder the computational time. In view of this, population chromosomes were chosen here to correspond to single path solutions comprising four transitional waypoints, excluding the start and end waypoints which are intialised and stored separately in the Mission Profile module. Each transitional waypoint is characterised by its three spatial coordinates X, Y, and Z. The latter were chosen to be 9-bit wide for the chomosomes and for the three-dimensional terrain map data stored separately in the Environment Profile module. The parent/offspring size and the number of transitional waypoints for all path solutions are not subjected to change during the entire evolutionary process. It was found through experimentation that a path solution with around 3 to 6 transitional waypoints provided good solutions. The bits encoding for the parameters of each chromosome and associated cost function (computed in the EM) is given in Table II.

2) FSM-based Approach: The sequential iterative process of the GA is mapped onto a parallel executed digital logic system by constituting sequential logic circuits through the use of finite state machines (FSMs). The FSM is established as the backbone of all the modules except the memory blocks. The state diagram of the CU is shown in Fig. 4, the rest of the modules are built on this principle.

3) 6-to-1 Multiplexer (MUX): The 6-to-1 MUX interface to the Parent and Offspring memories allows for all the modules to read the entire list of path solutions and overwrite the original contents of the underlying Block RAM with the list of updated path solutions. As compared to passing the entire population from module to module, the memory interface does

TABLE II
ENCODING OF EACH CHROMOSOME FOR THE PATH PLANNER

Parameter	Number of Bits	Integer Range
Transitional Waypoint $1.X$	9 Bits	$[0, 2^9 - 1]$
Transitional Waypoint $1.Y$	9 Bits	$[0, 2^9 - 1]$
Transitional Waypoint $1.Z$	9 Bits	$[0, 2^9 - 1]$
Transitional Waypoint $2.X$	9 Bits	$[0, 2^9 - 1]$
Transitional Waypoint $2.Y$	9 Bits	$[0, 2^9 - 1]$
Transitional Waypoint $2.Z$	9 Bits	$[0, 2^9 - 1]$
Transitional Waypoint $3.X$	9 Bits	$[0, 2^9 - 1]$
Transitional Waypoint $3.Y$	9 Bits	$[0, 2^9 - 1]$
Transitional Waypoint $3.Z$	9 Bits	$[0, 2^9 - 1]$
Transitional Waypoint $4.X$	9 Bits	$[0, 2^9 - 1]$
Transitional Waypoint $4.Y$	9 Bits	$[0, 2^9 - 1]$
Transitional Waypoint $4.Z$	9 Bits	$[0, 2^9 - 1]$
Evaluation Cost	32 Bits	$[0, 2^{32} - 1]$
Total Number of Bits	140 Bits	

require more clock cycles to retrieve population information. However, handling the population through a memory interface reduces the amount of resource utilisation. The tradeoff for additional clock cycle as to high resource utilisation was preferred.

4) Parent and Offspring Memories: The Parent and Offspring memories are implemented using the available FPGA Block RAM. As such, there is no shared memory external to the FPGA-based path planning system. As a compromise between available FPGA resources and model complexity, each of the Parent and Offspring memories was fixed to store 32 population chromosomes (i.e. path solutions).

5) Control Unit (CU): The CU operates as a finite state machine (FSM) that monitors the evolutionary process throughout the entire operation. The states and conditions are illustrated in Fig. 4.

6) Initial Population Module (IPM): Initially, the IPM generates 32 path solutions using randomly generated 9-bit binary strings to initialise each of the transitional waypoint components listed in Table II. Subsequently, when the reinitialise condition is met, the IPM then preserves the current best chromosome and regenerates 31 random path solutions. Note that the randomly generated initial population are generally zigzagged and overlapping in nature.

7) Selection Module (SM): Selection involves the identification of chromosomes from the Parent memory to undergo crossover and mutation. Tournament selection pressure is instituted with the fitter of two randomly chosen chromosomes flagged as the tournament winner. For this work, the SM is composed of 32 identical processing units which exploits the parallel processing capabilities of the FPGA by operating in parallel. Each unit randomly selects two chromosomes from the Parent memory for tournament selection where the winner is sent to the Offspring memory.

8) Genetic Operation Module (GOM): The genetic operation consists of two types of operators: crossover and mutation. For crossover, all transitional waypoints after a randomly chosen splitting point of the path are truncated and swapped between two randomly selected path solutions. This

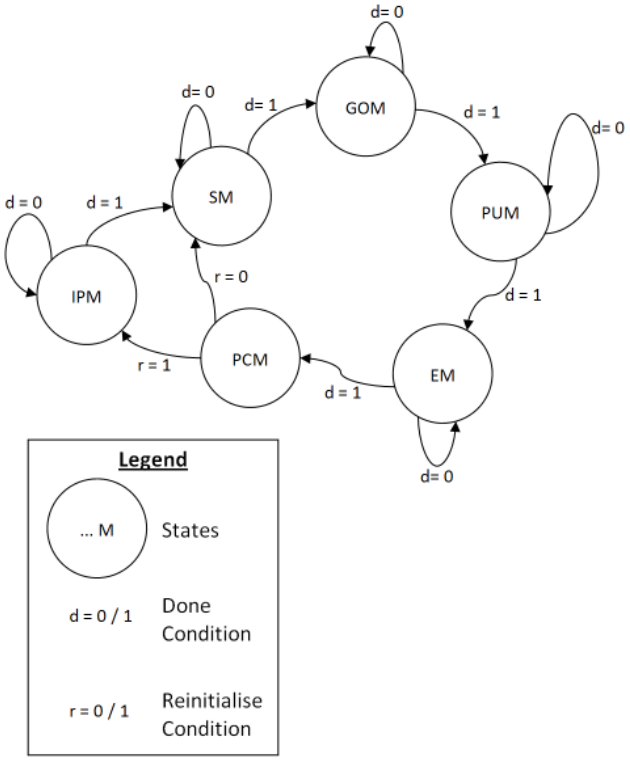

Fig. 4. State diagram of the Control Unit.

operation encourages exploitation of useful information from two chromosomes. The operation is illustrated in Fig. 5 for the present implementation involving four intermediate waypoints. For mutation, randomly selected path solutions are subjected to global perturb mutation, local perturb mutation, delete mutation, and swap mutation to promote exploitation of the search space with the expectation of speeding-up convergence to a global optimum (as represented in Fig. 6, Fig. 7, Fig. 8, and Fig. 9). The global and local perturb mutation are designed to induce small perturbation in a chromosome. Since the number of transitional waypoints is fixed at four, the delete mutation simply shifts a selected point spatially to the middle of a straight line connecting the preceding and following points. The swap mutation randomly picks two points and swaps their position.

For this work, the GOM is internally configured with crossover and mutation operations to be performed on all chromosomes of the Offspring memory. The GOM is composed of 24 identical crossover units (i.e. 75% of the offspring size) and eight mutation units (i.e. 25% of the offspring size). The eight mutation units consists of two global perturb mutation units, two local perturb mutation units, two delete mutation units, and two swap mutation units. Notably, all crossover and mutation operations for a single generation are conducted in parallel. The GOM generates a variety of parental combinations to produce 32 candidate path solutions.

9) Evaluation Module (EM): The fitness of each chromosome of the offspring is assessed based on contraints defined in the profiles such as feasibility and shortest distance. The EM evaluates the feasibility of the 32 new candidate path solutions

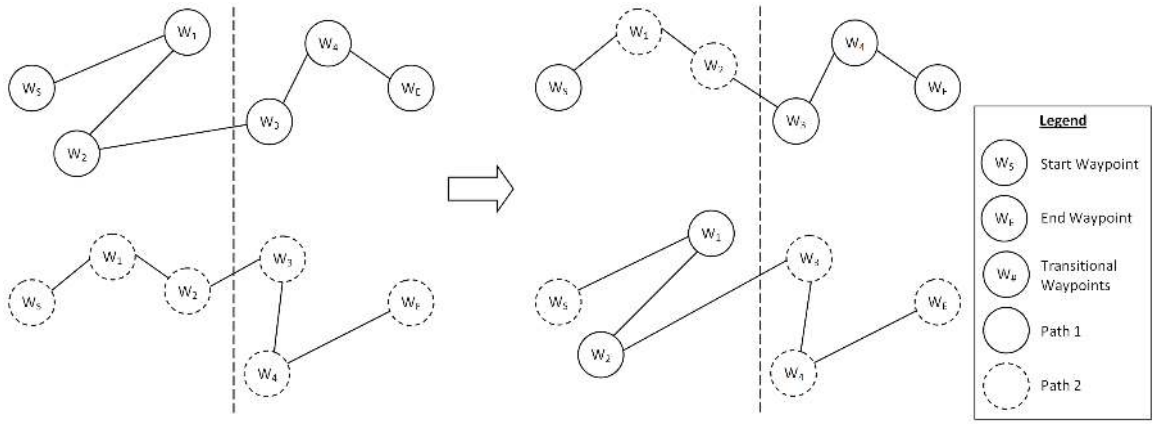

Fig. 5. General case for the crossover operation.

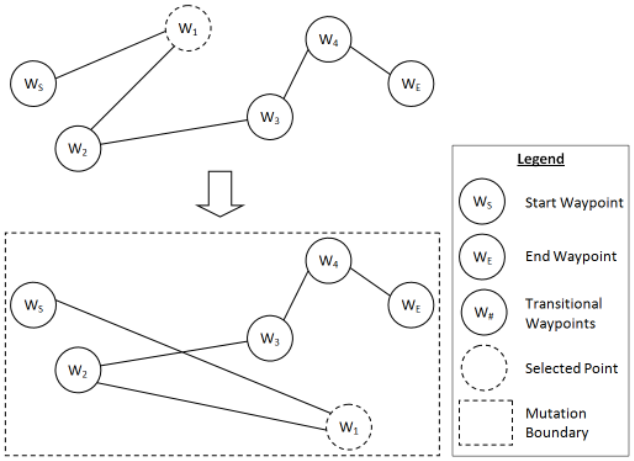

Fig. 6. General case for the global perturb operation.

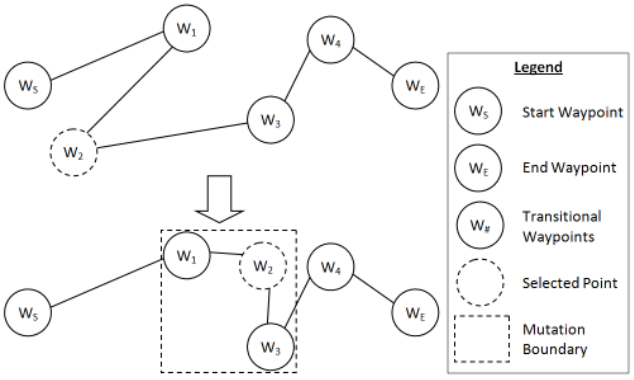

Fig. 7. General case for the local perturb operation.

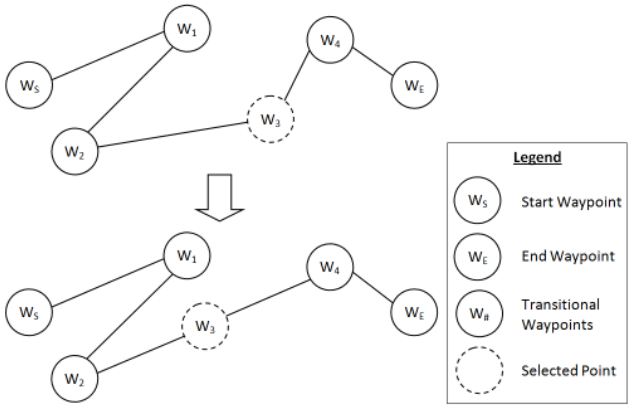

Fig. 8. General case for the delete operation.

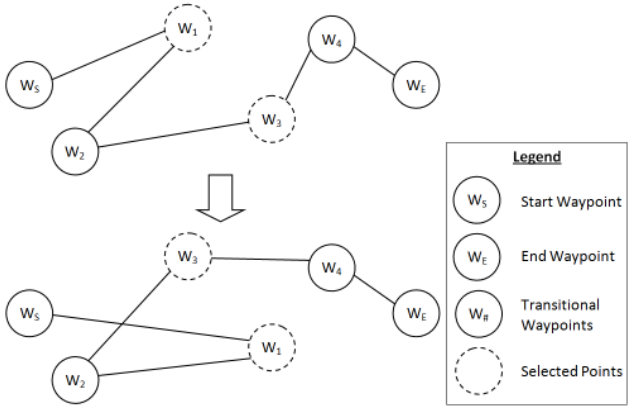

Fig. 9. General case for the swap operation.

and generates a new evaluation cost value for each one of them. The evaluation operations for a single generation are conducted in parallel. Additional constraint functions, such as vehicle kinematics, can be modularly included here into the EM. The inclusion of problem-specific constraints relating to the UAV, environment and mission profiles will be addressed below in the case studies.

10) Population Update Module (PUM): The population is updated using an elitist approach, where the selection of the best path solutions from the Parent and Offspring memories are retained and the remaining more inferior chromosomes are overwritten by the next generation.

The PUM concatenates the Parent and Offspring memories and sort them based on their fitness. Thereby, the new Parent memory that is ready for subsequent evolutionary

cycles consists a mix of surviving chromosomes from previous Parent memory and those which benefited from the genetic operations.

11) Premature Convergence Module (PCM): The PCM is made of comparators which monitor the Parent memory for premature convergence and triggers a reinitialise signal if necessary. Premature convergence occurs when there is a loss of genetic diversity within the population, causing the evolutionary search algorithm to get trapped in local optima in the early stages of the evolutionary process. The reinitialising of the Parent memory encourages the evolutionary process to recover with the reintroduction of new randomly generated population.

B. Synthesis Details

The design was synthesised the Xilinx ISE software with the Xilinx Virtex-4 LX200 as the target device, and the design goal was set to "balanced". A "balanced" design implies that no optimisation for speed or utilisation of FPGA resources was considered. Once the design was synthesised successfully, it was then compiled and built for implementation. This process consists of translating, mapping, placing and routing of the signals. For the design implementation process, no partition was specified and the design was translated and mapped successfully. All signals were placed and routed successfully as well, and all timing constraints were met. 32% of the logic slices on the device were utilised. The overall FPGA design had a maximum operating frequency of 83 MHz.

V. Case Study Experiments and Results

A. Experiments

Three case studies were performed to verify the effectiveness of the proposed FPGA-based path planning system. The first case study is on an empty environment profile, which tests the basics of the algorithm with the expectancy of a straight line prediction for the optimal path between the start and end waypoints. To further test and verify the predictive capabilities of the algorithm, the second and third case studies were on a simulation environment 512 m high by 512 m long by 512 m wide. The second case study did not set any maximum elevation in the mission profile, whereas the third constrained the UAV to a maximum elevation of 5 m below the highest peak. Five simulation runs are executed for each case study. A typical autopilot system has an update frequency of 10 Hz [38]. Therefore, it is essential for the time of convergence to be faster than the 100 ms threshold. For all three case studies, the start and end waypoints are kept to be identical. These examples are illustrative of an air sampling biosecurity mission or a remote sensing mission [39], [40].

The targeted UAV platform for the case studies was an autonomous helicopter measuring 0.5 m high and 1 m long (available from the ARCAA).The UAV profile was set to a rotary wing UAV so that kinematic constraints were assumed to be negligible.

The process used for generating the map data for use in the Environment Profile module is as follows. LIDAR (Light Detection And Ranging) technology provided the three-dimensional elevation map that was constructed into a lookup table (LUT) representation through the Sukharev grid [33] sampling theorem. The LUT can be seen as an X-by-Y matrix, with the referenced element corresponding to the elevation, Z, at that specific coordinate. The size of the LUT is 2.25 megabytes. The map data is within 1 m resolution, thereby each pixel is able to accommodate the rotary wing UAV without requiring any rescaling.

TABLE III
Encoding of Associated Cost

Parameter	Number of Bits	Integer Range
Collision Cost	8 Bits	$[0, 2^8 - 1]$
Distance Cost	24 Bits	$[0, 2^{24} - 1]$
Total Number of Bits	32 Bits	

For these case studies, a simple fitness function was used which incorporated a collision cost and distance cost only (see Table III). The former relates to the number of collision points along the path solution. The latter relates to the Euclidean distance of the path solution inclusive of the start and end waypoints. The cost returned by the fitness function was evaluated by concatenating the binary 8-bit representation of the collision cost before the 24-bit distance cost to arrive at a resultant 32-bit score.

B. Results

Fig. 10 shows the results for case study one with the expected results of a straight line. It took 18 generations for the algorithm to reach an absolute convergence, after which no better path solutions were found. Fig. 11 shows the results for case study two with the path initially climbing over the first mountainous region and subsequently curving towards the end waypoint. For this case study, the algorithm took 282 generations for convergence. Fig. 12 shows the results for case study three with maximum elevation included into the mission profile. The path solution now deviates around the mountainous region and continues towards the end waypoint. Convergence was achieved at 527 generations.

Table IV shows the results of the average computational time required for convergence. From the results, it can be seen that the algorithm takes longer to converge as the complexity of the task increases. Note that the output of the path planning architecture, a series of transitional waypoints, is intended to be coupled to an autopilot system via an appropriate data transformation, although that has not been implemented here. A typical autopilot has an update frequency of 10 Hz, thus it is essential that the results from the path planning module in all case studies must be output under 100 ms. All case studies are able to meet the 100 ms threshold.

VI. Conclusion

In this paper an autonomous GA-based UAV path planner is developed with all modules of the proposed architecture being entirely implemented on a single FPGA, including terrain data and flight constraints stored on the FPGA in separate UAV,

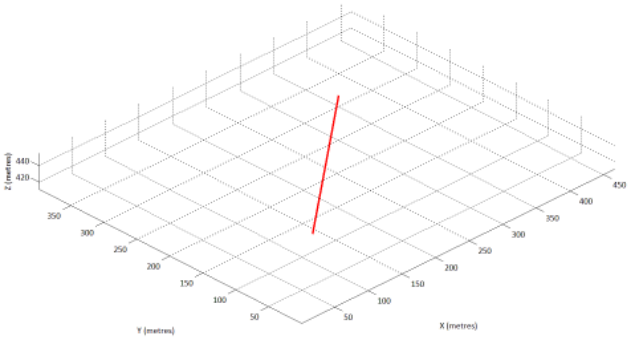

Fig. 10. Case study one.

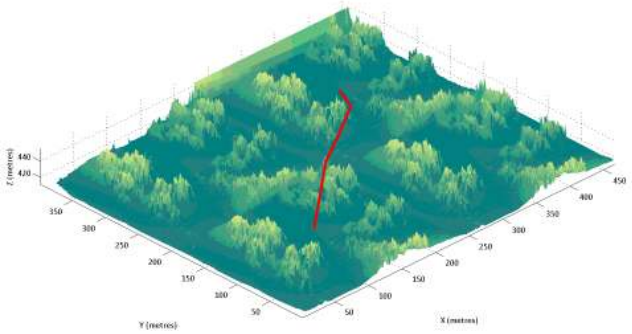

Fig. 11. Case study two.

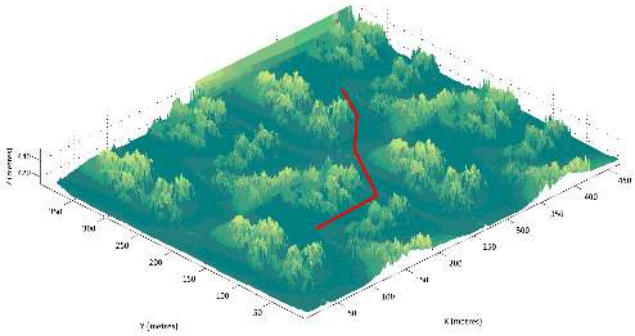

Fig. 12. Case study three.

TABLE IV
COMPUTATIONAL TIME RESULTS FOR THE CASE STUDIES

Case Study	Generation At Convergence	Average Convergence Time
Case Study One	18	2.7 ms
Case Study Two	282	21 ms
Case Study Three	527	47 ms

environment and mission profiles. The design architecture, operation, execution flow and communication are described in detail, as well as implementation issues arising from the choice of FPGA configurable hardware. Implementation targeting a Xilinx Virtex 4 FPGA development platform was achieved for a population size of thirty two, where each chomosome corresponds to a single path solution with four transitional waypoints between the start and end waypoints. Results of case studies for a small 1m long UAV helicopter with a 512 m^3 environment derived from LIDAR data demonstrate the ability of the FPGA-based path planner to generate feasible solutions with performance that can meet the 10 Hz update frequency of a typical autopilot system. The present work supports the use of an FPGA as suitable platform for flight constrained autonomous UAVs. While this work is focused on autonomous UAV applications, our approach may have the potential to be more widely applicable in hardware systems that also require on-board, real-time optimisation techniques.

Future work will involve testing the limitations and boundaries by, for example, increasing the complexity of the UAV and mission profiles (e.g. fixed-wing UAV kinematics, airspace restriction, and number of transitional waypoints), so that the whole FPGA-based UAV path planning algorithm can be verified as suitable for real-world applications. Also for consideration in future work is an examination of the reconfiguration capabilities of the FPGA, which could allow inclusion of other functions required for fully autonomous operation without compromising on-board path planning capabilities.

ACKNOWLEDGMENT

Computational resources and services used in this work were provided by the High Performance Computing and Research Support Group, Division of TILS, Queensland University of Technology, Brisbane, Australia. The authors would like to acknowledge the Cooperative Research Centre for Spatial Information (CRCSI) for their help with the LIDAR data used in this work and the support of the CRC for National Plant Biosecurity.

REFERENCES

[1] J. Kok, L. F. Gonzalez, R. Walker, N. Kelson, and T. Gurnett, "A synthesizable hardware evolutionary algorithm design for unmanned aerial system real-time path planning," in *Proceedings of the 2010 Australasian Conference on Robotics and Automation*, Brisbane, Australia, December 2010.

[2] Army UAS Center of Excellence Staff, "U.S. army roadmap for unmanned aircraft systems 2010-2035," U.S. ARMY, 2010.

[3] T. Cox, "Civil UAV capability assessment," NASA, Techincal report, 2004.

[4] M. R. Media, "U.S. military unmanned aerial vehicles market forecast 2010-2015," Research Report, 2010. [Online]. Available: http://www.marketresearchmedia.com/2010/04/09/unmanned-aerial-vehicles-uav-market/

[5] H. Shi, X. Sun, C. Sun, D. Chen, and Y. An, "Research of the path planning complexity for autonomous mobile robot under dynamic environments," in *Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications, 2006. ISDA '06*, vol. 3, October 2006, pp. 216–219.

[6] J. Leal, "Stochastic environment representation," Ph.D. dissertation, Australian Centre for Field Robotics, University of Sydney, Sydney, Australia, 2003.

[7] I. A. McManus, "A multidisciplinary approach to highly autonomous UAV mission planning and piloting for civilian airspace," Ph.D. dissertation, Cooperative Research Centre for Satellite Systems, Queensland University of Technology, Brisbane, Australia, 2005.

[8] B. J. Capozzi, "Evolution-based path planning and management for autonomous vehicles," Ph.D. dissertation, University of Washington, 2001.

[9] I. K. Nikolos, K. P. Valavanis, N. C. Tsourveloudis, and A. N. Kostaras, "Evolutionary algorithm based offline/online path planner for UAV navigation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 33, no. 6, pp. 898–912, December 2003.

[10] C. Zheng, L. Li, F. Xu, F. Sun, and M. Ding, "Evolutionary route planner for unmanned air vehicles," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 609–620, August 2005.

[11] C. W. Zheng, M. Y. Ding, and C. P. Zhou, "Real-time route planning for unmanned air vehicle with an evolutionary algorithm," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 17, no. 1, pp. 63–81, February 2003.
[12] D. Jia and J. Vagners, "Parallel evolutionary algorithms for UAV path planning," in *Proceedings of the AIAA 1st Intelligent Systems Technical Conference*, Chicago, Illinois, September 2004.
[13] G. Erinc and S. Carpin, "A genetic algorithm for nonholonomic motion planning," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, Roma, Italy, April 2007, pp. 1843–1849.
[14] C. Cocaud, "Autonomous tasks allocation and path generation of UAV's," Master's thesis, University of Ottawa, 2006.
[15] F. C. J. Allaire, M. Tarbouchi, G. Labonte, and G. Fusina, "FPGA implementation of genetic algorithm for UAV real-time path planning," *Journal of Intelligent and Robotic Systems*, vol. 54, no. 1-3, pp. 495–510, March 2009.
[16] P. H. W. Leong, "Recent trends in FPGA architectures and applications," in *Proceedings of the 4th IEEE International Symposium on Electronic Design, Test and Applications*, Hong Kong, January 2008, pp. 137–141.
[17] L. Vacariu, F. Roman, M. Timar, T. Stanciu, R. Banabic, and O. Cret, "Mobile robot path-planning implementation in software and hardware," in *Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation*, 2007.
[18] B. Girau and A. Boumaza, "Embedded harmonic control for dynamic trajectory planning on FPGA," in *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*, 2007, pp. 244–249.
[19] N. Sudha and A. R. Mohan, "Design of a hardware accelerator for path planning on the euclidean distance transform," *Journal of Systems Architecture*, vol. 54, pp. 253–264, 2008.
[20] T. K. Priya, P. R. Kumar, and K. Sridharan, "A hardware-efficient scheme and FPGA realization for computation of single pair shortest path for a mobile automaton," *Microprocessors and Microsystems*, vol. 30, pp. 413–424, 2006.
[21] O. Hachour, "A genetic_FPGA algorithm path planning of an autonomous mobile robot," in *Proceedings of the 10th WSEAS International Conference on Mathematical Methods, Computational Techniques and Intelligent Systems*, 2008, pp. 66–71.
[22] H.-C. Huang, C.-C. Tsai, and S.-C. Lin, "SoPC-based parallel elite genetic algorithm for global path planning of an autonomous omnidirectional mobile robot," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2009. SMC 2009.*, October 2009, pp. 1959–1964.
[23] M. Schmidt and D. Fey, "An optimized FPGA implementation for a parallel path planning algorithm based on marching pixels," in *Proceedings of the 2010 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, December 2010, pp. 442–447.
[24] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*. New York: MIT Press, 1990.
[25] S. Akishita, S. Kawamura, and K. Hayashi, "Laplace potential for moving obstacle avoidance and approach of a mobile robot," in *Proceedings of the Japan-USA Symposium on Flexible Automation, A Pacic Rim Conference*, 1990, pp. 139–142.
[26] J. O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 338–349, 1992.
[27] N. Sudha, S. Nandi, and K. Sridharan, "Cellular architecture for euclidean distance transformation," *Computers and Digital Techniques*, vol. 147, pp. 335–342, 2000.
[28] S. Morgan, "A comparison of simplex method algorithms," Master's thesis, University of Florida, 1997.
[29] T. Lozano-Perez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, pp. 560–570, 1979.
[30] I. Al-Taharwa, A. Sheta, and M. Al-Weshah, "A mobile robot path planning using genetic algorithm in static environment," *Journal of Computer Science*, vol. 4, pp. 341–344, 2008.
[31] D. Fey and D. Schmidt, "Marching pixels: a new organic computing principle for high speed CMOS camera chip," in *Proceedings of the ACM*, 2005, pp. 1–9.
[32] J. Barraquand, B. Langlois, and J. C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, pp. 224–241, 1992.
[33] A. G. Sukharev, "Optimal strategies of the search for an extremum," *U.S.S.R. Computational Mathematics and Mathematical Physics*, vol. 11, pp. 910–924, 1971.
[34] P. E. Danielssona, "Euclidean distance mapping," *Computer Graphics and Image Processing*, vol. 14, pp. 227–248, 1980.
[35] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs*, 3rd ed. Springer-Verlag, 1996. [Online]. Available: http://www.loc.gov/catdir/enhancements/fy0815/95048027-d.html
[36] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Boston, MA, USA: Addison-Wesley Professional, 1989.
[37] M. Serra, T. Slater, J. C. Muzio, and D. M. Miller, "The analysis of one-dimensional linear cellular automata and their aliasing properties," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 7, pp. 767–778, July 1990.
[38] Y. Kang and J. Hedrick, "Linear tracking for a fixed-wing UAV using nonlinear model predictive control," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1202–1210, September 2009.
[39] L. F. Gonzalez, D.-S. Lee, and R. A. Walker, "Optimal mission path planning (MPP) for an air sampling unmanned aerial system," in *Proceedings of the 2009 Australasian Conference on Robotics and Automation*, S. Scheding, Ed. Sydney: Australian Robotics and Automation Association, 2009, pp. 1–9.
[40] F. Gonzalez, P. P. Narayan, M. P. Castro, and L. Zeller, "Development of an autonomous unmanned aerial system to collect time-stamped samples from the atmosphere and localize potential pathogen sources," *Journal of Field Robotics*, vol. 28, no. 6, pp. 961–976, October 2011, special Issue: Safety, Security, and Rescue Robotics.

Jonathan Kok is a PhD candidate affiliated with the Australian Research Centre for Aerospace Automation (ARCAA) and the School of Engineering Systems at Queensland University of Technology (QUT). His research focuses on optimisation problems and algorithms, specifically involving evolutionary algorithms and FPGAs.

Luis Felipe Gonzalez is a Lecturer at Queensland University of Technology (QUT) and the Australian Research Centre for Aerospace Automation (ARCAA). He has a degree in Mechanical Engineering. He completed his PhD on Multidisciplinary Design Optimization methods for UAV systems at The University of Sydney in 2005. Felipe joined ARCAA in 2006 and has directed his attention to enabling technologies for flight control and optimization of civilian UAVs systems. He has developed 6 operational UAVs and has written 18 journal papers and more than 35 peer reviewed papers in the topic of evolutionary algorithms and optimization.

Neil Kelson is User Services Manager of QUT's HPC & Research Support Group. This role involves extensive consultation/collaboration with researchers and research groups regarding effective utilisation of advanced computing tools and technologies. He has a PhD in computational science and also holds degrees in Mathematics, Education and Information Technology. Neil's technology and research interests are focussed on engineering computational fluid dynamics, code optimisation and parallelisation, and the use of FPGAs for high performance embedded and supercomputing applications.