Journal of Marine Science and Technology

Volume 25 | Issue 4 Article 10

INTELLIGENT COMPUTATION AND DSP-BASED LANDING CONTROL

Jih-Gau Juang
Department of Communications, Navigation and Control Engineering, National Taiwan Ocean University, Keelung, Taiwan, R.O.C.

Hou-Kai Chiou
Department of Communications, Navigation and Control Engineering, National Taiwan Ocean University, Keelung, Taiwan, R.O.C.

Chia-Ling Lee
Department of Communications, Navigation and Control Engineering, National Taiwan Ocean University, Keelung, Taiwan, R.O.C.

Follow this and additional works at: https://jmstt.ntou.edu.tw/journal

 Part of the Engineering Commons

Recommended Citation

Juang, Jih-Gau; Chiou, Hou-Kai; and Lee, Chia-Ling (2017) "INTELLIGENT COMPUTATION AND DSP-BASED LANDING CONTROL," *Journal of Marine Science and Technology*: Vol. 25: Iss. 4, Article 10.
DOI: 10.6119/JMST-017-0329-2
Available at: https://jmstt.ntou.edu.tw/journal/vol25/iss4/10

This Research Article is brought to you for free and open access by Journal of Marine Science and Technology. It has been accepted for inclusion in Journal of Marine Science and Technology by an authorized editor of Journal of Marine Science and Technology.

Journal of Marine Science and Technology, Vol. 25, No. 4, pp. 442-457 (2017)
DOI: 10.6119/JMST-017-0329-2

INTELLIGENT COMPUTATION AND DSP-BASED LANDING CONTROL

Jih-Gau Juang, Hou-Kai Chiou, and Chia-Ling Lee

Key words: PID control, evolutionary computation, DSP, automatic landing control, fuzzy CMAC, turbulence.

ABSTRACT

This paper presents several digital signal processor (DSP) based intelligent controllers for aircraft automatic landing systems (ALSs). Proportional-Integral-Derivative (PID) control law is adopted in the intelligent controller design. A fuzzy cerebellar model articulation controller (CMAC) is utilized to compensate for the PID control signal. Control gains are selected by evolutionary computation. The controllers' tracking performance of preset landing paths and capability to adaptively respond to different disturbances are demonstrated through hardware simulations. Different evolution methods, namely Adewuya crossover, arithmetical crossover, average crossover, convex crossover and blend crossover are utilized to analyze the controllers' performances in terms of optimal parameter search. Hardware implementation of this intelligent controller is performed by a DSP board with a VisSim platform. This study also compares different CMACs in order to improve the performance of conventional ALSs. It is known that atmospheric disturbances affect not only the flight qualities of an aircraft, but also the flight safety. However, the proposed intelligent controllers can successfully expand the controllable conditions, even with severe wind disturbances.

I. INTRODUCTION

Evolutionary computation (EC) is the general term for several computational techniques based on the evolution of biological life in the natural world. In computer science, evolutionary computation is a subfield of artificial intelligence involving combinatorial optimization problems. Evolutionary computation includes genetic algorithm (GA), evolutionary programming (EP), evolution strategies (ES), genetic programming (GP), ant colony optimization (ACO), particle swarm optimization (PSO), classifier systems (CS) etc. In aircraft flight control problems, EC has been applied to path planning and landing control for decades. PSO was applied to an integrated landing reference system, which guaranteed the security margin and deflection margin (Zhang and Wang, 2013). Juang et al. (2004) also used GA without quantization to design a longitudinal control system based on a fuzzy controller to improve the precision of trajectory tracking and the security of landing for unmanned aerial vehicle (UAV) flight control. The designed automatic landing control system demonstrated good robust performance with wind disturbance, perturbation of aerodynamic parameters and sensor measurement errors. Imae et al. (1999) applied GP to deal with the flight control problem of a passenger aircraft encountering wind shear during the landing process. Based on GP, they proposed a method to construct flight controllers with perturbation performance, even when encountering wind shear, and demonstrated its effectiveness using Boeing-727 type aircraft. In Kanury and Song (2006), a flight management system that not only generates optimal trajectory but also produces smooth control action for path tracking was proposed for multiple aerial vehicles. GA was used to generate the optimal flight path when the target positions were given with unknown obstacles. A memory-based control strategy was developed to steer the vehicle along the generated trajectory, which can be dynamically adjusted according to varying flight conditions. In Heimes (2001), EC was applied to the design of a full-envelope flight control system. A generic control law structure with generic gain schedule algorithm was defined. The EC algorithm searched for an optimum control law that was valid throughout the flight envelope. This approach eliminated the normal manual iterative development methodology used to develop aircraft control laws. It also provided a rapid prototyping capability that allowed an existing flight control design to be customized for different aircraft with different evolutionary strategies, mutation methods, selection methods and recombination operators. In Arantes et al. (2015), a multi-population GA for UAV path planning in emergency situations was presented. Its objective was to minimize damage and increase safety during landing. The authors proposed two strategies for planner systems based on a multi-population GA and a greedy heuristic. The FlightGear simulator was used to illustrate the UAV's behavior when landing under different wind velocities. GA has been used for a wide range of applications, as well as for specific applications focused on a specific requirement. Thus far many new methods have focused on mak-

Paper submitted 11/10/14; revised 04/27/16; accepted 03/19/17. Author for correspondence: Yuchen Wang (e-mail: wobushiwangyuchen@live.cn).
Department of Communications, Navigation and Control Engineering, National Taiwan Ocean University, Keelung, Taiwan, R.O.C

ing GA more efficient, and on increasing the width of the parameter search ability. This study focuses on the crossover principle of different ECs. Five crossover principles (Michalewicz, 1992; Eshelman and Schaffer, 1993; Adewuya, 1996) are used with intelligent controllers under wind disturbances to search optimal control gains, and the different performances of the principles are compared.

According to a report by Boeing (Boeing Publication, 2000), the primary cause of 67% of all air accidents is human error, and 5% are caused by weather factors. By flight phase, 47% accidents occur during final approach or landing. It is therefore desirable to develop an intelligent automatic landing system (ALS) that expands the operational envelope to include safer responses under a wider range of conditions. In this study, the robustness of the proposed controller is obtained by choosing optimal control gains that allow a wide range of disturbances to the controller. The goal of this paper is to show that the proposed intelligent ALS can reduce reliance on human operators and guide the aircraft to a safe landing in severe turbulence environments. This study first uses a conventional automatic landing control system that uses a PID controller with EC as the adjustment mechanism in order to improve the performance of the conventional ALS and guide the aircraft to a safe landing. Wind disturbances are also included in the flight simulations. Many researchers have applied intelligent concepts to the problem of intelligent landing control, but some studies are not adaptive to various wind disturbance conditions (Nho and Agarwal, 2000; Heimes et al., 2001; Ionita and Sofron, 2002; Kanury and Song, 2006). In past decades, most improvements to the ALS system have been for the guidance instruments, such as Global Navigation Satellite System (GNSS) Integrity Beacons, Global Positioning Systems, Microwave Landing Systems, and Automatic Land Position Sensors (Cohen et al., 1995; DDC-I, 1995; Kaufmann and McNally, 1995; Asai et al., 1997). By using improvement calculation methods and high accuracy instruments, these systems provide more accurate flight data to the ALS, allowing a smoother landing. However, these researches do not include weather factors such as wind turbulence. Recently, intelligent concepts such as neural networks, fuzzy systems, genetic algorithm, and hybrid systems have been applied to flight control to increase the flight controller's adaptive capability to different environments (Jorgensen and Schley, 1991; Izado et al., 1998; Juang and Lin, 2008; Juang et al., 2008). In studies preceding this paper (Juang and Chin, 2004; Juang and Lin, 2008; Juang et al., 2008; Juang et al., 2011), the authors applied a multilayered fuzzy modeling neural network as a controller (Juang and Chin, 2004). Control gains of pitch autopilot were selected by GA. The linguistic fuzzy controller was tuned by back propagation through a time learning scheme, which could overcome turbulence of up to 75 ft/sec. In (Juang and Lin, 2008), the authors used an intelligent control scheme that integrated a cerebellar model articulation controller (CMAC) and genetic algorithms (GA) in automatic landing control, to make automatic landing systems more intelligent. The fuzzy CMAC was applied as a compensator for the PID control. GA was also used in searching optimal control gains of the pitch autopilot. The performance of the fuzzy-CMAC was demonstrated to be more robust than a conventional CMAC, and it was able to safely guide an aircraft through turbulence of up to 90 ft/sec. In Juang et al. (2011), different neural network controllers were applied to automatic landing control design. The back propagation network, multi-functional link network, counter propagation network, improved back propagation network and radial basis function network, were able to overcome turbulence of up to 65 ft/sec, 65 ft/sec, 30 ft/sec, 55 ft/sec and 30 ft/sec, respectively. An adaptive resource allocating network (ARAN) was also proposed to improve the performance of conventional ALS. Adaptive learning rates were obtained through analysis of Lyapunov stability in order to guarantee the learning convergence. The ARAN controller was able to overcome turbulence of up to 75 ft/sec. In Juang et al. (2008), the authors proposed an intelligent automatic landing controller that uses recurrent neural networks (RNN) with genetic algorithms (GAs) to improve the performance of conventional ALS. The conventional PID controller was replaced by the RNN controller with optimal control gains in an environment with atmospheric turbulence. Real-time recurrent learning (RTRL) was applied to train the RNN, which used gradient-descent of the error function with respect to the weights to perform the weights updates. The control scheme utilized five crossover methods of GAs to search optimal feedforward and feedback control gains of the pitch autopilot, with different turbulence strengths. For the Adewuya, Arithmetical, Average, Convex and Blend GAs, maximal turbulences were 90 ft/sec, 85 ft/sec, 85 ft/sec, 85 ft/sec and 85 ft/sec, respectively Here, a type-2 FCMAC (Liu et al., 2007; Wang et al., 2004; Liang and Mendel, 2000) was used to improve the performance of conventional ALS. Comparisons of conventional CMAC (Albus, 1975) and conventional (type-1) FCMAC (Juang and Lin, 2008) were also given. The performance of the intelligent ALS in extreme environmental conditions can be improved by the advantages of the CMAC, which include local generalization and a rapid learning process. Meanwhile, this study also utilizes VisSim software and TI C2000 Rapid Prototyper to develop an embedded control system that uses a DSP controller. Thus, hardware-in-the-loop control can be achieved.

II. LANDING SYSTEM

At the aircraft landing phase, the pilot descends from the cruise altitude to an altitude of approximately 1200 feet above the ground. The pilot then positions the aircraft so that the aircraft is on a heading towards the runway centerline. When the aircraft approaches the outer airport marker, which is about 4 nautical miles from the runway, the glide path signal is intercepted, as shown in Fig. 1 (Juang et al., 2011). As the airplane descends along the glide path, its pitch, attitude and speed must be controlled. The descent rate is about 10 ft/sec, and the pitch angle is between -5 to +5 degrees. Finally, as the airplane descends to 20 to 70 feet above the ground, the glide path control system is disengaged and a flare maneuver is executed. The

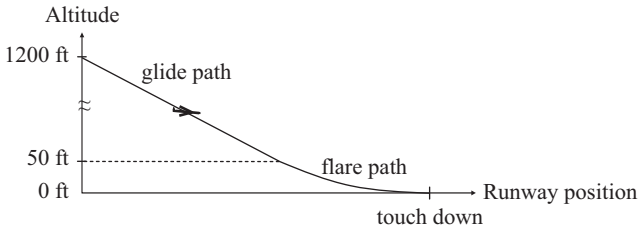

Fig. 1. Glide path and flare path.

vertical descent rate is decreased to 2 ft/sec so that the landing gear is able to dissipate the energy of the impact at landing. The pitch angle of the airplane is then adjusted, to between 0 to 5 degrees for most aircraft, which allows a soft touchdown on the runway surface.

A simplified model of a commercial aircraft that moves only in the longitudinal and vertical plane is used in the simulations for implementation ease (Jorgensen and Schley, 1991). The motion equations of the aircraft are given as follows:

$$\dot{u} = X_u(u-u_g) + X_w(w-w_g) + X_q\Delta q - g(\frac{\pi}{180})\cos(\gamma_0)\Delta\theta + Z_E\delta_E + Z_T\delta_T \quad (1)$$

$$\dot{w} = Z_u(u-u_g) + Z_w(w-w_g) + (Z_q - \frac{\pi}{180}U_0)\Delta q - g(\frac{\pi}{180})\sin(\gamma_0)\theta_T \quad (2)$$

$$\dot{q} = M_u(u-u_g) + M_w(w-w_g) + M_q\Delta q + M_E\delta_E + M_T \quad (3)$$

$$\dot{\theta} = q \quad (4)$$

$$\dot{h} = -w + \frac{\pi}{180}U_0\theta \quad (5)$$

where u is the aircraft's longitudinal incremental velocity (ft/sec), w is the aircraft's vertical incremental velocity (ft/sec), q is the pitch rate (rate/sec), θ is the pitch angle (deg), h is the aircraft's altitude (ft), δ_E is the incremental elevator angle (deg), δ_T is the throttle setting (ft/sec), γ_o is the flight path angle (-3deg), and g is the gravity (32.2 ft/sec^2). The parameters X_i, Z_i and M_i are the stability and control derivatives.

In order to make the ALS more intelligent, reliable wind profiles are necessary. Two spectral turbulence form models by von Karman and Dryden are mostly used for aircraft response studies. In this study the Dryden form (Jorgensen and Schley, 1991) was used for its demonstration ease. The model is given by:

$$u_g = u_{gc} + N(0,1)\sqrt{\frac{1}{\Delta t}}\left(\frac{\sigma_u\sqrt{2a_u}}{s+a_u}\right) \quad (6)$$

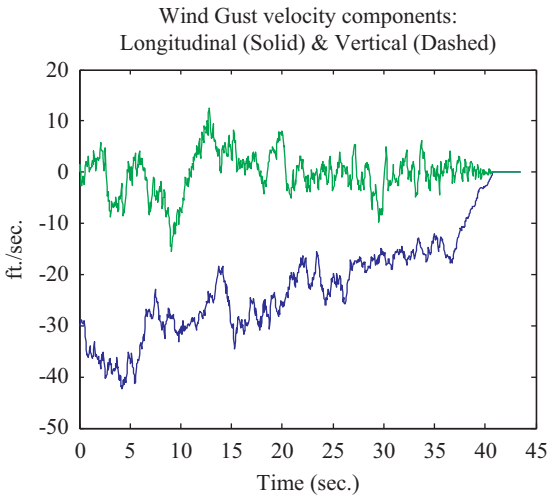

Fig. 2. Turbulence profile.

$$w_g = N(0,1)\sqrt{\frac{1}{\Delta t}}\left(\frac{\sigma_w\sqrt{3a_w}(s+b_w)}{(s+a_u)^2}\right) \quad (7)$$

where $u_{gc} = -u_{wind510}[1 + \frac{\ln(h/510)}{\ln(51)}]$, $a_u = \frac{U_o}{L_u}$, $L_w = h$, $a_u = \frac{U_o}{L_u}$, $a_w = \frac{U_o}{L_w}$, $b_w = \frac{U_o}{L_w\sqrt{3}}$, $L_u = 100h^{1/3}$ for $h > 230$, $L_u = 600$ for $h \le 230$, $\sigma_w = 0.2|u_{gc}|(0.5 + 0.00098\times h)$ for $0 \le h \le 500$, $\sigma_w = 0.2|u_{gc}|$ for $h > 500$. The parameters are: u_g is the horizontal wind velocity (ft/sec), w_g is the vertical wind velocity (ft/sec), U_0 is the nominal aircraft speed (ft/sec), $u_{wind510}$ is the wind speed at 510 ft altitude, L_u and L_w are scale lengths (ft), σ_u and σ_w are RMS values of turbulence velocity (ft/sec), Δt is the simulation time step (sec), $N(0, 1)$ is the Gaussian white noise with zero mean and unity standard deviation, u_{gc} is the constant component of u_g, and h is the aircraft altitude (ft). Fig. 2 shows a turbulence profile with a wind speed of 30 ft/sec at 510 ft altitude (Juang et al., 2011).

III. CONTROL SCHEME

A conventional aircraft landing system uses PID-type control, as shown in Fig. 3 (Juang et al., 2011). Controller inputs consist of altitude and altitude rate commands along with aircraft altitude and altitude rate. The pitch command θ_c is obtained from the PID controller. Then, the pitch autopilot is controlled by pitch command. The pitch autopilot is shown in Fig. 4 (Juang et al., 2011). In order to enable aircraft to land more stably once they have entered the flare path, a constant pitch angle will be added to the controller. In general, the PID controller is simple and effective, but there are some drawbacks such as apparent

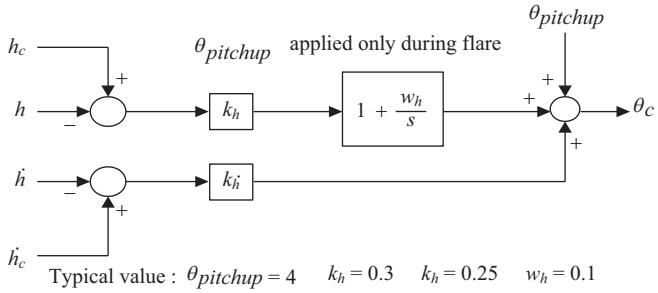

Fig. 3. PID-controller.

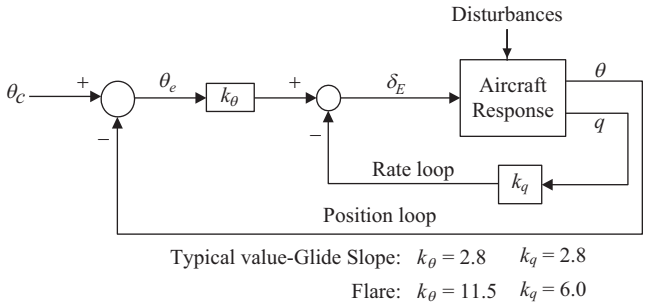

Fig. 4. Pitch autopilot.

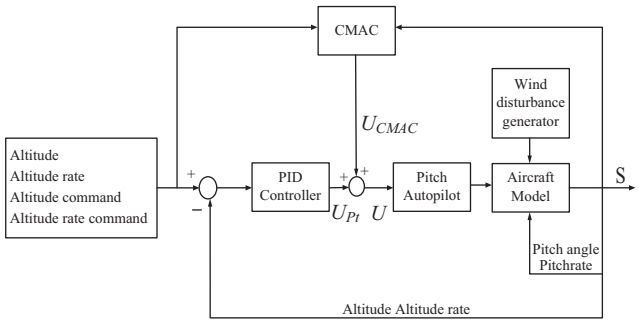

Fig. 5. The CMAC control scheme.

overshoot and sensitivity to external noise and disturbance. When severe turbulence is encountered the PID controller may not be able to guide the aircraft to a safe landing. With the CMAC compensator, the proposed controller can overcome these disadvantages. It uses a traditional PID controller to stabilize the system and train the CMAC to provide precise control. The original gains of the PID controller are adjusted based on experience, and what it provides are tolerable solutions, rather than desired solutions. The CMAC can effectively ameliorate these conditions.

The overall control scheme is described in Fig. 5 (Juang and Lin, 2008), in which the control signal U is the sum of the PID controller output and the CMAC output. The inputs for the CMAC and PID controller are: altitude, altitude command, altitude rate and altitude rate command. In each time step k, the CMAC involves a recall process and a learning process. In the recall process, it uses the desired system output of the next time step and

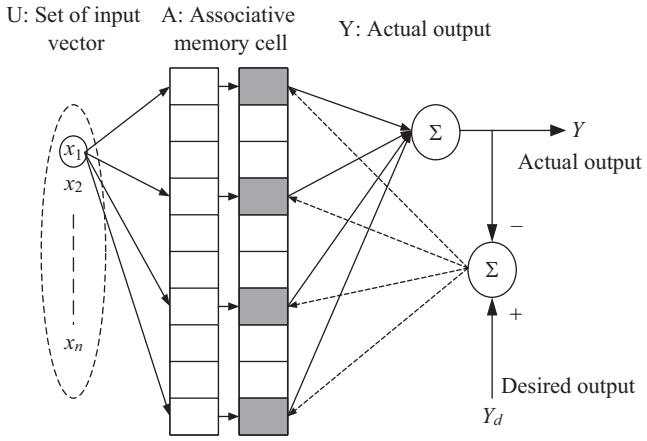

Fig. 6. Conceptual diagram of CMAC.

the actual system output as the address to generate the control signal U_{CMAC}. In the learning process, the control signal of the pitch autopilot U is treated as a desired output. It is used to modify the weights of CMAC stored at a location addressed by the actual system output and the system output of the next time step. The output of the CMAC is the compensation for pitch command. When the wind turbulence is too strong, the ALS cannot achieve a safe landing. Here, CMAC, type-1 FCMAC, and type-2 FCMAC control schemes are used to improve the turbulence resistance of the ALS.

1. Cerebellar Model Articulation Controller (CMAC)

CMAC is a type of artificial neural network proposed in (Albus, 1975). It could be considered an associative memory learning structure based on the performance of the human cerebellum. The function of CMAC is akin to a lookup-table technique which represents complex and nonlinear systems. The fundamental concept of CMAC is to store information into overlapping regions in an associative approach so that stored information can be easily recalled using less storage space (memory cell). The structure of CMAC is shown in Fig. 6 (Juang and Lin, 2008). Manipulation of the CMAC divides the algorithm into two segments.

First is the output generating stage. The output of CMAC can be obtained by the mapping process $U \to A \to Y$, where A represents the M-dimensional memory cell, the $a \in A \subset R_M$ is the binary associative vector, as an address indexes in coherence with the input vector x. Let the input x address N ($N < M$) memory cells; the mapping $A \to Y$ represents the chosen weights stored in memory cells are added together to compute the output as:

$$y(x) = \sum_{j=1}^{N} w_j a_j(x) \tag{8}$$

where w_j is the weight of the j^{th} storage hypercube and $a_j(x)$ is a binary factor indicating whether the j^{th} storage hypercube is addressed by the input x.

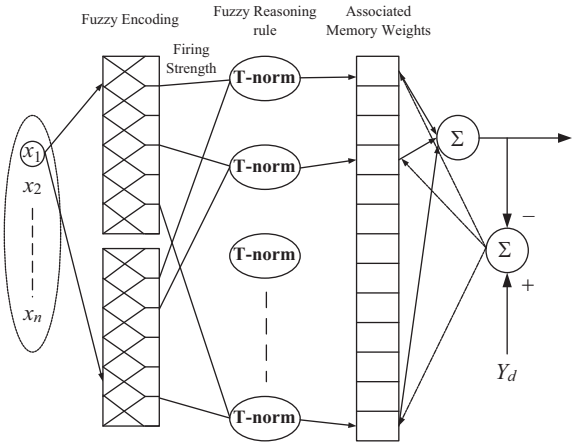

Fig. 7. conventional FCMAC structure.

The second stage is the CMAC network learning stage, which serves to update the addressed memory cell weights according to the error between the desired output and the real output. Its weight updating rule is:

$$w_j^{(i)} = w_j^{(i-1)} + \frac{\alpha}{m}(y_d - \sum_{j=1}^{N} w_j^{(i-1)} a_j) \tag{9}$$

where y_d is the desired output, m is the number of addressed memory cells and α is the learning rate.

When the CMAC input vector is processed, it is simply divided into certain blocks. The relation between the input vector and these blocks is a crisp relation. The relation between the input condition and the association intensity is simply "activated" or "not activated". Furthermore, an important CMAC identity is local generalization derived from where nearby input vectors overlap and share some associative memory cells.

2. Type-1 FCMAC

The structure of type-1 FCMAC is shown in Fig. 7. FCMAC is a kind of associative memory network. It not only has a faster self learning rate than normal neural networks by quantities with a few adjustments of memory weights, but also has good local generalization ability. The function of FCMAC is similar to a look-up table, and the output of CMAC is figured from a linear combination of weights stored in memory. The concept of FCMAC is to store data (knowledge) in overlapped storage hypercubes (remembering the region) in an associative manner such that the stored data can easily be recalled. Two kinds of operations are included in the FCMAC: one is the calculation of the output result and the other is learning and adjusting the weight. The output of FCMAC can be obtained by the mapping process $X \to S \to C \to W \to Y$ as follows.

Step 1: Quantization ($X \to S$): X is n-Dimension Input Space.

For the given $x = [x_1\, x_2, \ldots, x_n]^T$, $s = [s_1\, s_2, \ldots, s_n]^T$ represents the quantization vector of x. The corresponding state of each input variable is specified before fuzzification.

Step 2: Associative Mapping Segment ($S \to C$):

This step fuzzifies the quantization vector quantized from x. FCMAC uses the fuzzification method of the fuzzy theorem as its addressing scheme. After the input vector is fuzzified, the input state values are transformed to "firing strength", which is based on corresponding membership functions.

Step 3: Memory Weight Mapping ($C \to W$):

After fuzzifying block regions, the i^{th} rule's firing strength in FCMAC could be computed as:

$$C_j(x) = c_{j1}(x_1) * c_{j2}(x_2) * \ldots c_{jn}(x_n) = \prod_{i=1}^{n} c_{j1}(x_i) \tag{10}$$

where $c_{ji}(x_i)$ is the j^{th} membership function of the i^{th} input vector, and n is the total number of states. The asterisk "*" denotes a fuzzy t-norm operator. There are several kinds of t-norms, such as the max, min and product operators. This study chooses the product inference method as the t-norm operator because it is easy to implement.

Step 4: Output Generation with Memory Weight Learning ($W \to Y$):

Because of the partial proportional fuzzy rules and overlap situation, multiple fuzzy rules are fired simultaneously. The consequences of these multi-rules are merged by a defuzzification process. The defuzzification approach applied in this study sums the assigned weights of the activated fuzzy rules based on their firing strengths, denoted as $c_j(x)$. The network output is:

$$y = \sum_{j=1}^{N} (w_j C_j(x) / \sum_{i=1}^{N} C_i(x)) \tag{11}$$

The FCMAC learning serves to update the memory weight according to the error between the desired output and the actual output. The weight update rule for FCMAC is as follows (Juang and Lin, 2008):

$$w_j^{(i)} = w_j^{(i-1)} + \frac{\alpha}{m}(y_d - y) C_j(x) / \sum_{i=1}^{N} C_i(x) \tag{12}$$

where α is the learning rate, m is the floor size (called generalization) and y_d is the desired output.

3. Type-2 Fuzzy CMAC

The type-2 fuzzy theorem is used in the CMAC structure in order to promote more accurate resolution than conventional FCMAC. The mapping procedure of type-2 FCMAC is similar to that of conventional FCMAC. The diagram structure of type-2 FCMAC is shown in Fig. 8. Each mapping phase is described as follows. X is an n-dimensional input space, as shown in Fig. 9. For a given $X = [x_1\, x_2, \ldots, x_n]$, $S = [s_1\, s_2, \ldots, s_n]$ represents the quantization vector of x. The corresponding state of each input variable is specified before fuzzification. Type-2 FCMAC uses

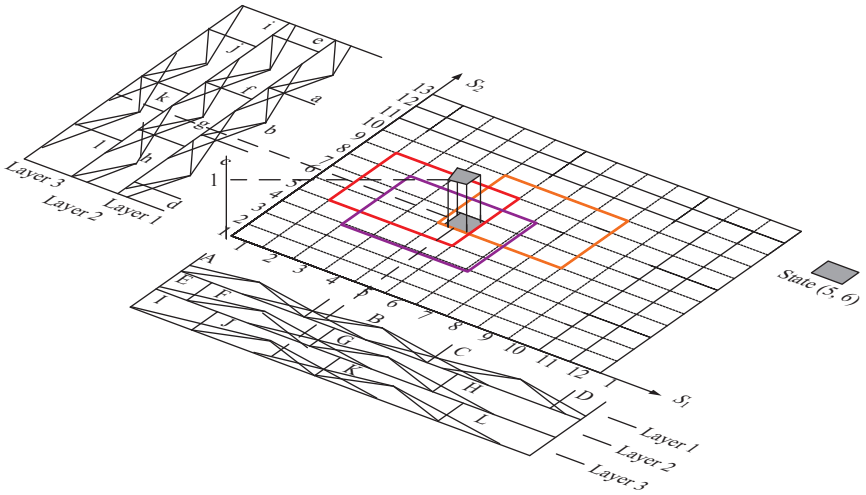

Fig. 8. Diagram of type-2 FCMAC in 3-D.

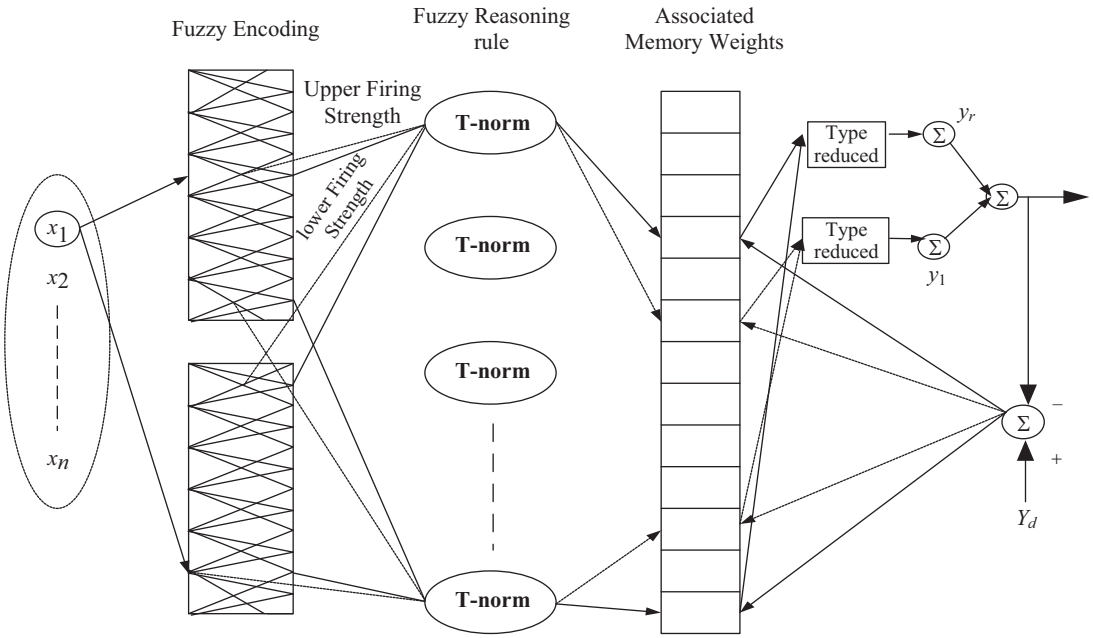

Fig. 9. Architecture of type-2 FCMAC network.

the interval type-2 fuzzification method of the fuzzy theorem as its addressing scheme. After the input vector to the interval type-2 fuzzy set is fuzzified, the input state values are transformed to upper firing strength and lower firing strength, which is based on corresponding interval type-2 membership functions. This study chooses the product inference method as the t-norm operator. The j^{th} rule's upper firing strength $\overline{c}^j$ and lower firing strength c^j in type-2 FCMAC could be computed as:

$$\overline{c}^j(x) = \overline{c}_{j_1}(x_1) * \overline{c}_{j_2}(x_2) * \ldots \overline{c}_{j_n}(x_n) = \prod_{i=1}^{n} \overline{c}_{j_i}(x_i) \quad (13)$$

$$\underline{c}^j(x) = \underline{c}_{j_1}(x_1) * \underline{c}_{j_2}(x_2) * \ldots \underline{c}_{j_n}(x_n) = \prod_{i=1}^{n} \underline{c}_{j_i}(x_i) \quad (14)$$

The type-reduced set of the type-2 FCMAC using the center of sets type reduction is:

$$y_{\cos} = [y_l, y_r] = \int_{w^1 \in [\underline{w}^1, \overline{w}^1]} \cdots \int_{w^N \in [\underline{w}^M, \overline{w}^N]} \cdots \int_{c^1 \in [\underline{c}^1, \overline{c}^1]} \cdots \int_{c^M \in [\underline{c}^M, \overline{c}^N]} 1 \Big/ \frac{\sum_{j=1}^{n} c^j w^j}{\sum_{j=1}^{n} c^j}. \quad (15)$$

This is an interval type-1 set determined by its left and right end points y_l and y_r, which can be written as follows (Liang and Mende, 2000):

$$y_r = \frac{\sum_{j=1}^{N} \overline{c}^j \overline{w}^j}{\sum_{j=1}^{N} \overline{c}^j} = \frac{\sum_{j=1}^{R} \underline{c}^j \overline{w}^j + \sum_{j=R+1}^{N} \overline{c}^j \overline{w}^j}{\sum_{j=1}^{R} \underline{c}^j + \sum_{j=R+1}^{N} \overline{c}^j} \tag{16}$$

$$y_l = \frac{\sum_{j=1}^{N} \underline{c}^j \underline{w}^j}{\sum_{j=1}^{N} \underline{c}^j} = \frac{\sum_{j=1}^{L} \overline{c}^j \underline{w}^j + \sum_{j=L+1}^{N} \underline{c}^j \underline{w}^j}{\sum_{j=1}^{L} \overline{c}^j + \sum_{j=L+1}^{N} \underline{c}^j} \tag{17}$$

where $\overline{w}$ and $\underline{w}$ are the corresponding weights of $\overline{c}$ and c, respectively. L and R can be obtained from (Liang and Mendel, 2000):

Step 1: Assume that the pre-computed $\overline{w}^j$ are arranged in ascending order, i.e.,

$$\overline{w}^1 \le \overline{w}^2 \le \ldots \le \overline{w}^N .$$

Step 2: Compute y_r by initially setting $\overline{c}^j = (\overline{c}^j + \underline{c}^j)/2$ for $j = 1 \ldots N$ and let $y_r' = y_r$.

Step 3: Find $R(1 \le R \le N-1)$ such that $\overline{w}^R \le y_r' \le \overline{w}^{R+1}$.

Step 4: Compute y_r with $\overline{c}^j = \underline{c}^j$ for $j \le R$ and $\overline{c}^j = \overline{c}^j$ for $j > R$, and let $y_r'' = y_r$.

Step 5: If $y_r'' \ne y_r$ then go to step 6. If $y_r'' = y_r'$ then stop and set $y_r'' \equiv y_r'$.

Step 6: Set $y_r' = y_r''$ and return to Step 3.

The procedure for computing y_l is very similar to the one just given for y_r. In Step 3 find $L(1 \le L \le N-1)$ such that $\underline{w}^L \le y_l' \le \underline{w}^{L+1}$. Additionally, in Step 2 compute y_l initially setting $\underline{c}^j = (\overline{c}^j + \underline{c}^j)/2$ for $j = 1 \ldots N$ and in Step 4 compute y_l with $\underline{c}^j = \overline{c}^j$ for $j \le L$ and $\underline{c}^j = \underline{c}^j$ for $j > L$.

The defuzzified output is simply the average:

$$y = y_r + y_l \tag{18}$$

Type-2 FCMAC learning serves to update the memory weight according to the error between the desired output and the actual output. The learning rule for type-2 FCMAC is as follows:

$$\overline{w}_j^{(i)} = \overline{w}_j^{(i-1)} + \frac{\alpha}{m}(y_d - y)\overline{c}_j(x) / \sum_{j=1}^{N} \overline{c}_j(x) \tag{19}$$

$$\underline{w}_j^{(i)} = \underline{w}_j^{(i-1)} + \frac{\alpha}{m}(y_d - y)\underline{c}_j(x) / \sum_{j=1}^{N} \underline{c}_j(x) \tag{20}$$

where α is the learning rate and m is the floor size (called generalization).

IV. OPTIMAL CONTROL PARAMETERS

GA was first proposed by John Holland in 1962 (Holland, 1962), and is an optimization and search technique based on the principles of genetics and natural selection. In 1975, Holland mentioned the most basic principle of GA in *Adaptation in Natural and Artificial Systems* (Holland, 1962). In the same year, De Jong showed the usefulness of GA for function optimization, and made the first concerted effort to find optimized GA parameters. GA generally only involves techniques of implementing mechanisms, such as reproduction, crossover, mutation, fittest function, etc. via reproduction, crossover, and mutation steps, and is able to generate consequent generations to reach the purpose of the evolution. Based on a population's degree of fitness value, GA retains the fittest and eliminates inferior populations. GA has therefore been widely used to solve optimization problems. GA can search many points at the same time and is not prone to falling into local optima. In recent years, many researchers have improved the crossover and mutation mechanisms in GA in order improve its performance. In a previous study (Juang et al., 2008), the authors used different crossover methods in recurrent neural network controller designs, and obtained better performance than that achieved with the original control parameters. In this study, these crossover methods are applied to the ALS control scheme with fuzzy CMAC controller. Roulette wheel selection is applied in the reproduction process. Different crossover operations are given as follows.

1. Adewuya Crossover Method

The operation is divided into three steps, as shown below (Adewuya, 1996).

Step 1

Randomly choose a gene from each individual of a matching pair in parent generation, $P_{m\alpha}$ and $P_{n\alpha}$, as crossover site.

$$pattern_1 = \begin{bmatrix} p_{m1} & p_{m2} & \ldots & p_{m\alpha} & \ldots & p_{ms} \end{bmatrix} \tag{21}$$

$$pattern_2 = \begin{bmatrix} p_{n1} & p_{n2} & \ldots & p_{n\alpha} & \ldots & p_{ns} \end{bmatrix} \tag{22}$$

Step 2

Calculate new values of these selected genes as follows, where β is a random number and $0 \le \beta \le 1$.

$$p_{new1} = (1-\beta) \cdot p_{m\alpha} + \beta \cdot p_{n\alpha} \tag{23}$$

$$p_{new2} = \beta \cdot p_{m\alpha} + (1-\beta) \cdot p_{n\alpha} \tag{24}$$

Step 3

Replace $P_{m\alpha}$ and $P_{n\alpha}$ with P_{new1} and P_{new2}, respectively. The genes in the right side of the crossover site exchange with each other, which will result in new offspring.

$$Newpattern_1 = \begin{bmatrix} p_{m1} & p_{m2} & \cdots & p_{new1} & \cdots & p_{ns} \end{bmatrix} \quad (25)$$

$$Newpattern_2 = \begin{bmatrix} p_{n1} & p_{n2} & \cdots & p_{new2} & \cdots & p_{ms} \end{bmatrix} \quad (26)$$

2. Arithmetical Crossover Method

The arithmetical crossover makes the mating pair move apart or draw together (Michalewicz, 1992).

$$\text{Move apart} \quad \begin{aligned} x_1^{'} &= x_1 + \sigma \cdot (x_1 - x_2) \\ x_2^{'} &= x_2 - \sigma \cdot (x_1 - x_2) \end{aligned} \quad (27)$$

$$\text{Draw together} \quad \begin{aligned} x_1^{'} &= x_1 + \sigma \cdot (x_2 - x_1) \\ x_2^{'} &= x_2 - \sigma \cdot (x_2 - x_1) \end{aligned} \quad (28)$$

where x_1 and x_2 are the parents, $x_1^{'}$ and $x_2^{'}$ are the new offspring, and σ is a random and positive small real value. In addition, either (27) or (28) can be used with -1 < σ < 1, which will determine whether the mating pair moves apart or draws together by the sign of σ.

3. Average Crossover Method

The average crossover method uses a simplified model with (23) and (24) where β is 1/2. It can be obtained as follows:

$$p_{new} = \frac{1}{2} \cdot (p_{m\alpha} + p_{n\alpha}) \quad (29)$$

where $P_{m\alpha}$ and $P_{n\alpha}$ are the parents and P_{new} is the new offspring.

4. Convex Crossover Method

The convex crossover is shown below:

$$x_{new} = \gamma \cdot x_j + (1 - \gamma) \cdot x_k \quad (30)$$

where x_j and x_k are the parents, x_{new} is the new offspring and γ is a random and small real value.

5. Blend Crossover Method

The blend crossover (BLX-α) was proposed by Eshelman and Schaffer (Eshelman and Schaffer, 1993). It is a prominent crossover operator for GA, and excels in optimization of a number of standard separable functions with multimodality. The BLX-α crossover generates offspring using the following operation:

$$X_i^1 = \min(x_i^1, x_i^2) - \alpha \cdot d_i \quad (31)$$

$$X_i^2 = \max(x_i^1, x_i^2) + \alpha \cdot d_i \quad (32)$$

$$d_i = \left| x_i^1 - x_i^2 \right| \quad (33)$$

where x^1 and x^2 are chosen randomly from the population, x_i^1 and x_i^2 are the i^{th} elements of x^1 and x^2, respectively. The value of each element x_i^c of the offspring vector x^c is uniformly sampled from the interval $\left[X_i^1, X_i^2\right]$. α is a positive parameter, with a suggested value of 0.5.

Finally, the mutation in GA is a very important process, as it permits the introduction of extra variability into the population. This study chooses a population at random, and changes its gene information. However, the new offspring must be in the range established after adding gene information. The real number muation process is applied as follows:

$$x_{new} = x_{old} + s \cdot rand\_noise \quad (34)$$

where s is the random value between 0 to 1. The fitness function that was used in this control scheme is:

$$\text{Fitness} = \text{number of successful landings with different turbulence strengths} \quad . \quad (35)$$

V. HARDWARE REALIZATION

VisSim is a Windows-based program for the modeling and simulation of complex nonlinear dynamic systems (Visual Solution, 2002). VisSim combines an intuitive drag & drop block diagram interface with a powerful simulation engine. The visual block diagram interface offers a direct method for constructing, modifying and maintaining system models. The simulation engine provides fast and accurate solutions for linear, nonlinear, continuous time, discrete time, time varying and hybrid system designs. This study used ViSim to build a dynamic aircraft model, and realized the conventional PID controller by the same manner. The intelligent controller design using C language and its realization by DSP are also presented (Juang et al., 2011).

Since the invention of the transistor and integrated circuit, digital signal processing functions have been implemented on many hardware platforms ranging from special-purpose architectures to general-purpose computers. It was not until all of the functionality (arithmetic, addressing, control, I/O, data storage and control storage) could be realized on a single chip that DSP could become an alternative to analog signal processing for the wide range of applications seen today. This study uses a TI TMS320LF2407 chip to perform the desired tasks. The 2407A devices offer the enhanced TMS320DSP architectural design of the C2xx core CPU for low-cost, low-power and high-

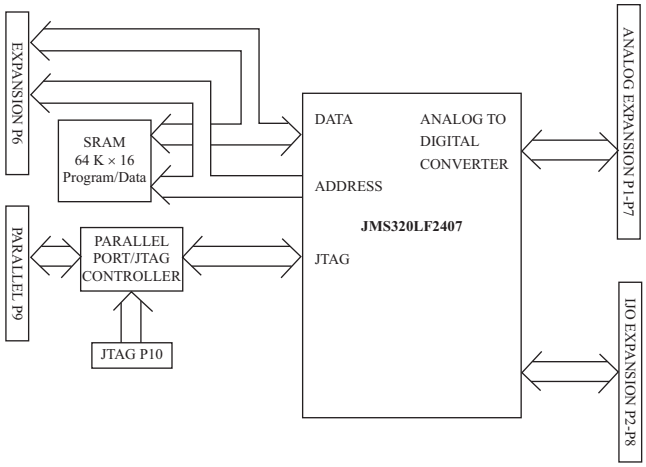

Fig. 10. Working process of the eZdspTMLF2407A board.

Fig. 11. Externals of the eZdspTMLF2407A board.

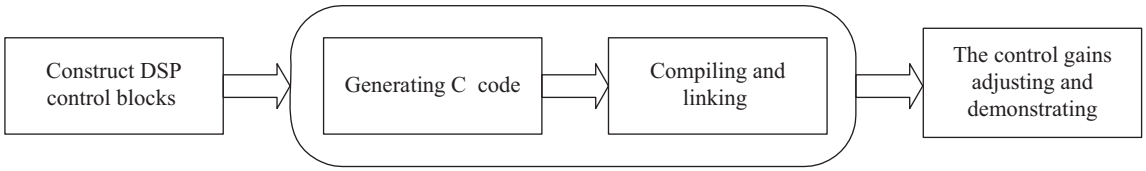

Fig. 12. Flow chart of VisSim/DSP procedure development.

performance processing capabilities. Moreover, it offers a suitable array of memory sizes and peripherals tailored to meet the specific performance points required by various applications. The TMS320LF2407 operates at 40 MHz (40 MIPS), has 4 to16 PWM output channels and has serial communication capabilities. In addition, the TMS320LF2407 contains a 10-bit analog-to-digital converter (ADC), with a minimum conversion time of 500 ns and up to 16 channels of analog input. Furthermore, the working process and externals of the eZdspTMLF2407A board are shown in Figs. 10 and 11, respectively.

There are three basics steps in the development of a DSP algorithm:

(1) Create the system to be executed on the target DSP;
(2) Generate the C source code from the system;
(3) Compile and link the C source code to produce an executable file.

If step 1 is performed using available blocks from VisSim software, steps 2 and 3 are automatically performed by VisSim. The core of the VS-ECD2407 is the TI TMS320LF2407A, which is a 16-bit fixed-point DSP. When designing the controller of an aircraft, the problem of the fixed point must be taken into account, because VisSim are molds all have floating-point operation. Thus, the VisSim/Fixed-Point software must be matched to the design molds of the fixed-point flight controller. Fig. 12 shows DSP development of the procedure entirely, which is divided into the following steps.

Step 1: In VisSim software, the fixed-point controller molds of an aircraft are designed by VisSim/TI C2000 Rapid Prototype and VisSim/Fixed Point.
Step 2: CCStudio can make fixed-point controller molds to perform compiling, analysis, debugging and demonstration, and generate *.c code and *.out code.

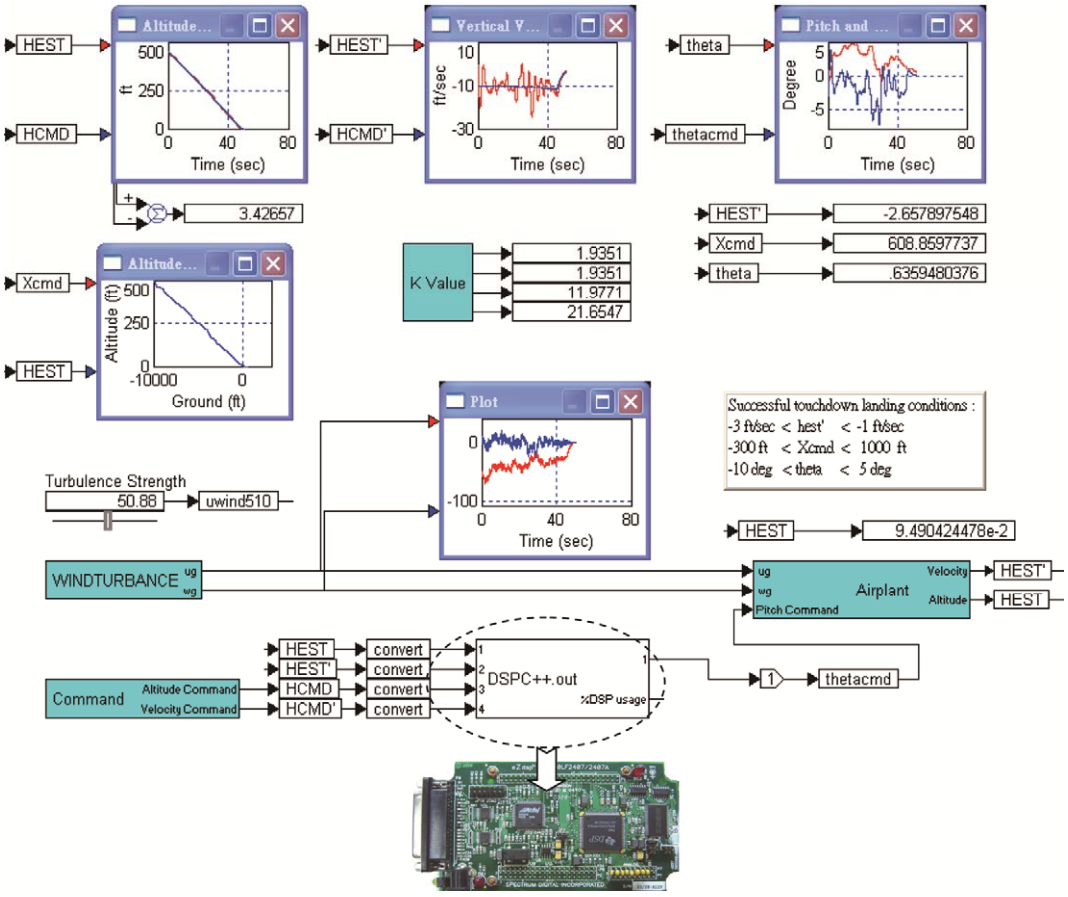

Fig. 13. DSP Hardware in-the-loop mode.

Step 3: Generate DSP controller molds which include *.out code, and replace original fixed-point controller molds.

Step 4: Download *.out code to TI TMS320LF2407A embedded flash memory by JTAG.

Step 5: Utilize DSP controller to control automatic landing system and show real-time relevant flight behavior.

By the above DSP controller development procedure, the entire real-time DSP hardware in-the-loop mode is shown in Fig. 13. From the VisSim development platform, it will transmit information, altitude, altitude rate, altitude and altitude rate commands to the VS-ECD2407 via a connecting JTAG between the computer and the VS-ECD2407. After DSP processing, it passes the pitch command and adjusts the angle of elevation back to the pitch autopilot in VisSim via JTAG. This enables an aircraft to follow a chosen landing trajectory. The advantages of DSP are its fast operation, powerful instruction, fixed addressing ability at high speed, parallel processing etc. These can significantly improve processing speed and accuracy, such that it can be applied to real-time control. In the VisSim simulation selection items, one can choose "Run in Real Time", and while designing fixed-point DSP controller, choose the exchange frequency of the datum between DSP and PC properly, then real-time control can be performed.

VI. SIMULATION RESULTS

The aircraft starts the initial states of the ALS as follows: the flight height is 500 ft, the horizontal position before touching the ground is 9240 ft, the flight angle is -3 degrees, and the speed of the aircraft is 234.7 ft/sec. Successful touchdown landing conditions are defined as follows:

(1) $-3 \le \dot{h}_{TD} \le -1$ (ft/sec)

(2) $-300 \le x_{TD}(T) \le 1000$ (ft)

(3) $200 \le V_{TD}(T) \le 270$ (ft/sec)

(4) $-10 \le \theta_{TD}(T) \le 5$ (degrees)

where T is the time at touchdown, $\dot{h}_{TD}$ is vertical speed, x_{TD} is the horizontal position, V_{TD} is the horizontal speed and θ_{TD} is the pitch angle.

1. Conventional PID Controller

Table 1 shows the results from using different wind turbulence speeds. The conventional PID controller with original control gains can only successfully guide an aircraft through wind speeds of 0 ft/sec to 30 ft/sec (Jorgensen and Schley, 1991). If the wind

Table 1. Results obtained using conventional PID controller (control gains: $K1 = 2.8$, $K2 = 2.8$, $K3 = 11.5$, $K4 = 6.0$).

Wind speed	Landing point (ft)	Aircraft vertical speed (ft/sec)	Pitch angle (degree)
0	797	-2.83	-1.41
10	910	-2.55	-0.85
20	809	-2.38	-0.59
30	844	-2.19	-0.17

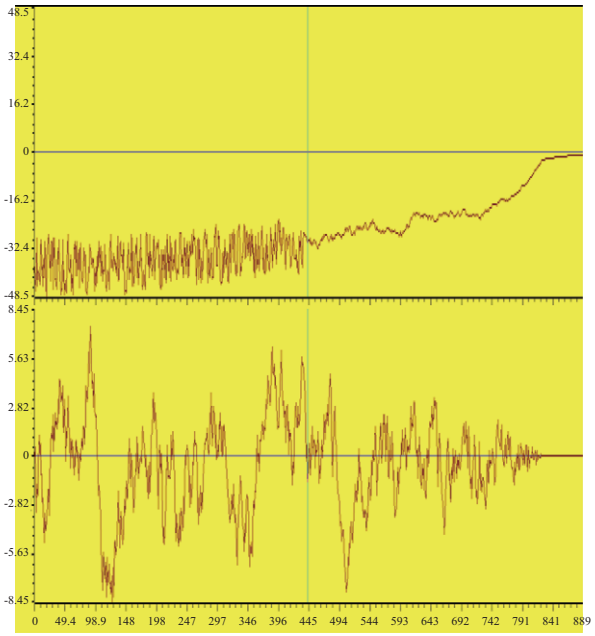

Fig. 14. Turbulence profile (30 ft/sec).

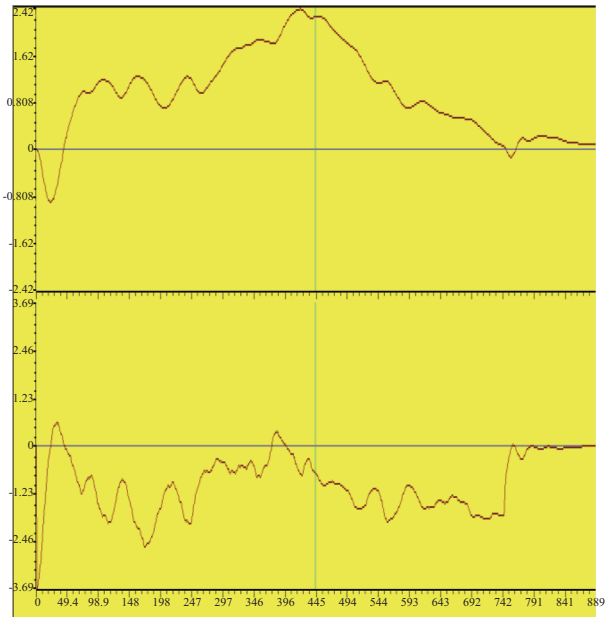

Fig. 15. Aircraft pitch (top) and command (bottom).

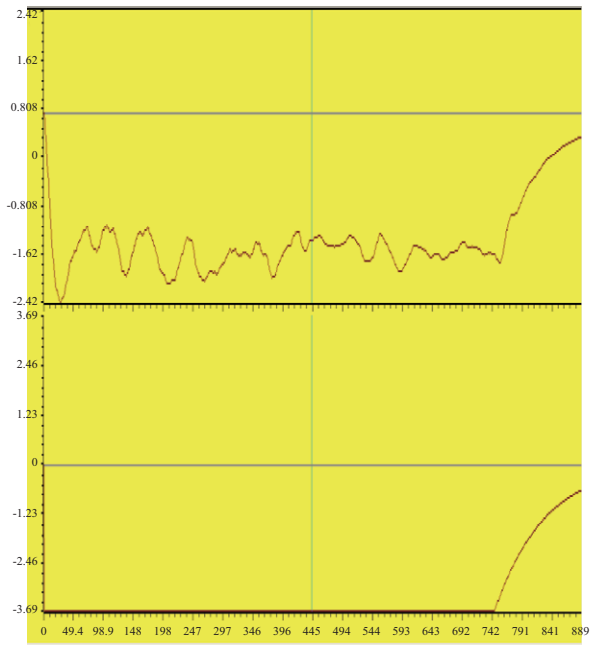

Fig. 16. Vertical velocity (top) and command.

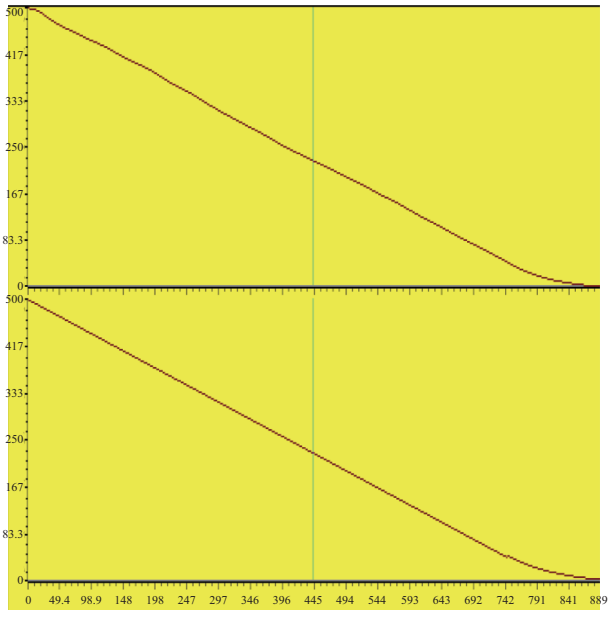

Fig. 17. Aircraft altitude (top) and command.

speed is higher than 30 ft/sec, the ALS will be unable to guide an aircraft to land safely. An aircraft can thus only achieve a safe landing in wind turbulence of up to 30 ft/sec, as shown in Figs. 14-17.

Table 2. Results obtained using CMAC controller.

Wind speed	Landing point (ft)	Aircraft vertical speed (ft/sec)	Pitch angle (degree)
0	854	-2.55	-0.96
10	762	-2.76	-0.93
20	774	-2.51	-0.61
30	844	-2.72	-0.41
40	691	-1.93	0.21
50	586	-2.26	0.87
58	844	-2.58	0.98

Table 3. Results obtained using type-1 FCMAC control.

Wind speed	Landing point (ft)	Aircraft vertical speed (ft/sec)	Pitch angle (degree)
10	797	-2.83	-1.41
30	938	-1.54	-0.58
50	891	-2.13	0.47
70	691	-2.21	1.41
90	926	-1.99	1.34

Table 4. Results obtained using type-2 FCMAC control.

Wind speed	Landing point (ft)	Aircraft vertical speed (ft/sec)	Pitch angle (degree)
20	855	-2.51	-0.58
40	726	-2.44	0.03
60	996	-2.00	0.50
80	890	-1.71	1.39
100	937	-2.21	2.05

Table 5. Fixed number of generations.

	Adewuya crossover	Arithmetical crossover	Average crossover	Convex crossover	Blend crossover
CPU times RGA/Total	2.04%	1.61%	1.51%	1.75%	1.62%
Error (%)	9.28%	9.04%	9.06%	9.00%	9.20%
Max intensity (ft/sec)	70	70	65	70	75

2. Different CMAC Controllers

The results obtained using the CMAC controller are given in Table 2. The type-2 FCMAC control scheme can successfully guide the aircraft flying through wind speeds of 0 ft/sec to 100 ft/sec, while the type-1 FCMAC can only offer safe guidance below 90 ft/sec [21], as shown in Table 3. Table 4 shows the results obtained using type-2 FCMAC. Scenarios with wind turbulence of 100 ft/sec include a pitch angle of 2.05 degrees, vertical speed of -2.21 ft/sec, horizontal velocity of 234.68 ft/sec, and horizontal position at touchdown is 937 ft.

3. Evolutionary Computation-Based Controller

In evolutionary computation, regardless of binary-coded type or real-valued type, many new methods have been proposed improve the evolution, offering better parameter search ability. This section focuses on the crossover method of real-valued genetic algorithm. First of all, the relevant initial parameters of the real-valued genetic algorithm are fixed, and evolution procedures, namely reproduction and mutation, are all unanimous. The flow chart of a complete evolution is shown in Fig. 18. This study uses five crossover principles to search optimal control gains separately for an aircraft using a CMAC controller in turbulence wind conditions, and compares differences between these methods. The number of generations of evolution was fixed in order to calculate the execution time (CPU time) required for each crossover method.

Tables 5 and 6 show the comparison results obtained using

Table 6. Required number of generations.

	Adewuya crossover	Arithmetical crossover	Average crossover	Convex crossover	Blend crossover
CPU times RGA/Total	1.84%	1.63%	1.61%	1.43%	1.94%
Error (%)	9.21%	9.27%	9.09%	9.10%	9.20%
Max intensity (ft/sec)	80	75	75	75	75
Required generations	23	18	35	31	25

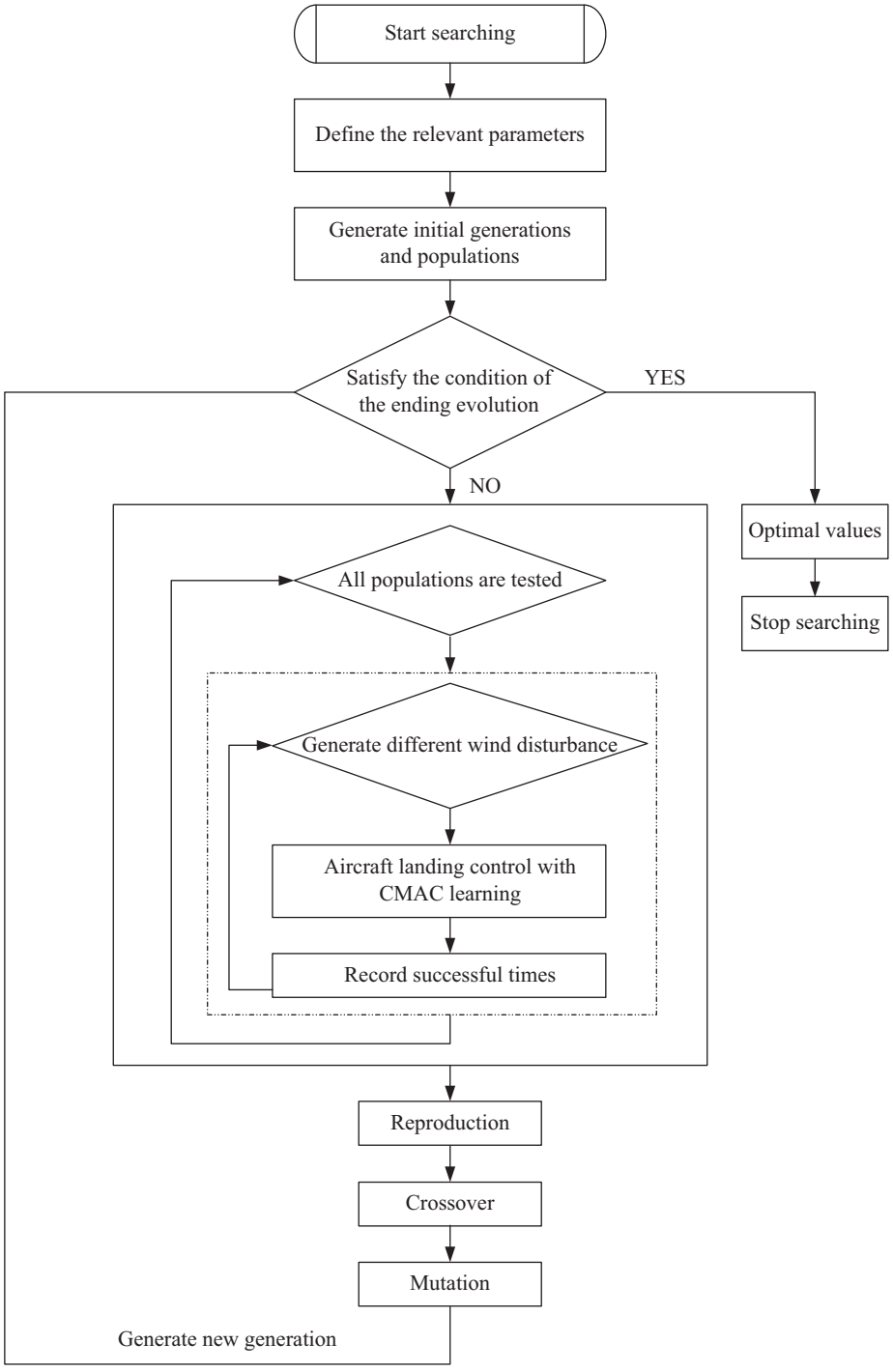

Fig. 18. Evolutionary learning process of the CMAC control scheme.

Table 7. Results obtained using type-1 FCMAC with optimal control gains in TMS320C6713 DSP board (optimal control gains: $K1 = 2.5409$, $K2 = 6.8029$, $K3 = 10.7398$, $K4 = 13.4932$).

Wind speed	Landing point (ft)	Aircraft vertical speed (ft/sec)	Pitch angle (degree)
10	785.2276	-2.6398	-1.0054
20	738.2920	-2.5844	-0.6968
30	861.7598	-2.4173	-0.2652
40	693.1877	-2.1031	0.3418
50	814.1143	-2.5232	0.3842
60	820.4292	-1.8753	1.0607
70	798.7611	-1.9576	1.3403
80	738.2920	-1.6419	2.0949
90	961.2381	-1.5932	1.6522
100	849.5021	-1.4839	2.0451
105	736.9072	-1.6998	2.5472

Table 8. Results obtained using type-2 FCMAC with optimal control gains in TMS320C6713 DSP board (optimal control gains: $K1 = 2.0738$; $K2 = 2.0738$; $K3 = 8.4802$; $K4 = 14.8921$).

Wind speed	Landing point (ft)	Aircraft vertical speed (ft/sec)	Pitch angle (degree)
10	713.4394	-2.7149	-1.0181
20	658.6411	-2.5085	-0.5884
30	689.9717	-2.5961	-0.3657
40	796.1005	-2.6263	0.0399
50	867.3648	-1.7812	0.6581
60	926.0343	-1.6648	0.7523
70	937.7682	-1.5684	1.0462
80	738.2920	-2.1779	1.3978
90	722.0064	-2.7847	2.5582
100	867.3648	-1.6653	2.1311
110	703.6141	-2.5363	2.2396
114	838.8158	-1.8290	2.9149

different crossover methods. All cases are tested by 10 independent runs. The results shown in Tables 5 and 6 are averaged values of these independent runs. This study fixed the number of generations to 10 in order to compare CPU times, as shown in Table 5. Table 6 shows the required number of generations until optimal control gains are obtained. From Table 5, the CPU time of the average crossover is the lowest. From the comparison of the error, which is the difference between altitude command and actual altitude under wind turbulence speeds at 70 ft/sec, the lowest value is achieved by the convex crossover. From Table 6, the Adewuya crossover can overcome the most intense wind turbulence, up to 80 ft/sec, and does not need more generations in convergence. In fact, the CPU times of these five crossover methods are similar, and only represent a very small portion of total execution time. Furthermore, the execution speeds of new computer technology is constantly and consistently improving. The execution time of the evolution is relative to the speed of the CPU and the amount of RAM used. Therefore, the difference lies in the parameters that are searched. Moreover, Tables 5 and 6 compare the results of percentage of CPU time, exhibiting only slight differences. This study also tested other application software for searching optimal control gains. Sharing part of the calculation resources clearly results in differences. In addition, by the change within unit time, wind turbulence is more violent than wind shear. Searching optimal control gains in wind turbulence is more difficult than in wind shear, and thus requires more generations. Since the Adewuya crossover method exhibited superior performance over the other methods tested, it is used in FCMAC control scheme. Tables 7 and 8 show the results obtained using type-1 FCMAC and type-2 FCMAC by Adewuya crossover method, respectively. Type-2 fuzzy CMAC with optimal control gains can enable the aircraft to automatically negotiate turbulence of up to 114 ft/sec, as shown in Figs. 19-22.

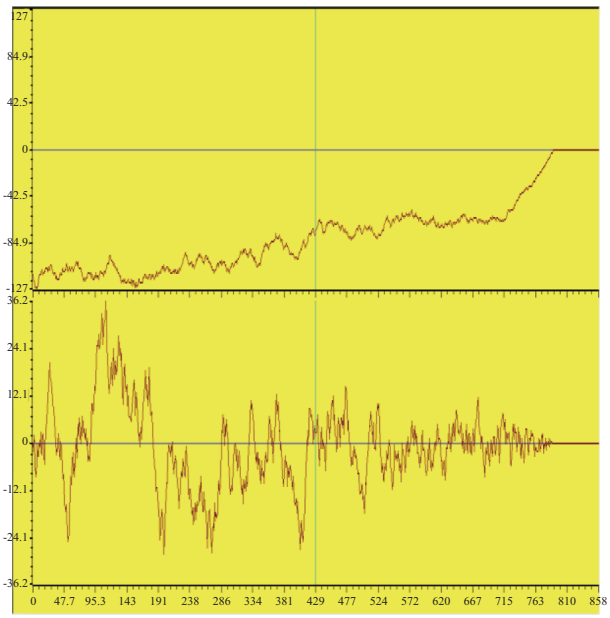

Fig. 19. Turbulence profile (100 ft/sec).

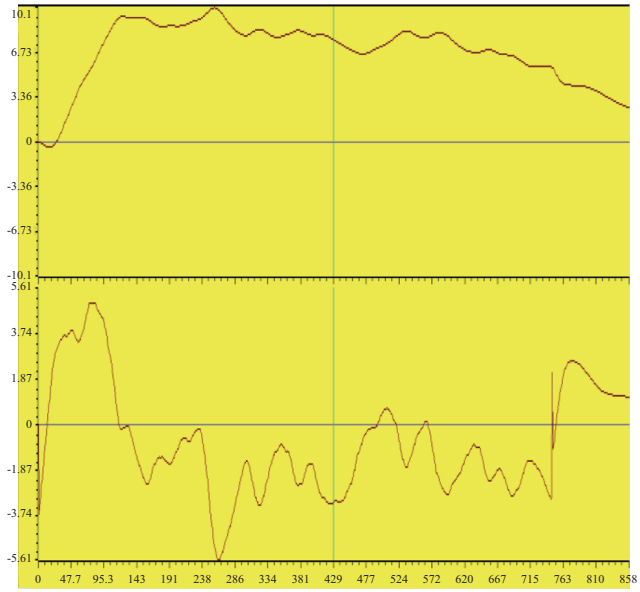

Fig. 20. Aircraft pitch (top) and command (bottom).

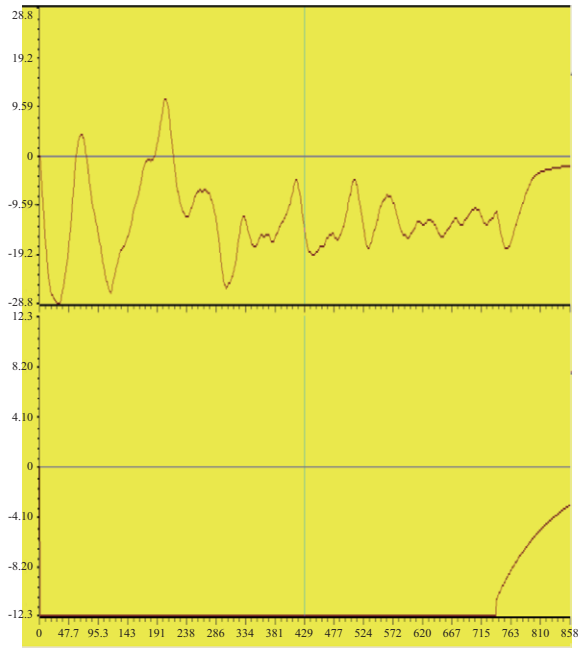

Fig. 21. Vertical velocity (top) and command.

Fig. 22. Aircraft altitude (top) and command.

VII. CONCLUSIONS

The purpose of this paper was to investigate the use of evolutionary computation and DSP with CMACs in aircraft automatic landing systems. CMAC is a non-fully connected associative memory network with overlapping receptive fields. Unlike the back propagation neural network which uses the global weight updating rule, CMAC is distinguished by the constant local weight updating rule. CMAC not only combines the advantages of rapid convergence speed and low computation, but is also easily realized by hardware. The conventional CMAC uses constant binary or triangular receptive-field basis functions. The characteristics of fuzzy systems are human-like reasoning and expert knowledge. Therefore, fuzzy CMAC includes the learning ability of neural networks and the advantages of fuzzy systems. A fuzzy system that uses type-2 fuzzy sets and type-2 fuzzy logic and inference is called a type-2 fuzzy system. In traditional fuzzy system models, the structure is characterized by using type-1 fuzzy sets. Type-1 fuzzy sets, defined on a universe of discourse, map an element of the universe of discourse onto a crisp number in the unit interval [0, 1]. However, type-2 fuzzy can translate the linguistic and numerical uncertainty from

original data into fuzzy rule uncertainty, while the type-1 cannot. Type-2 fuzzy sets provide better ability to handle uncertainty than type-1 fuzzy sets. The type-2 FCMAC is a generalization of the feed-forward neural network, fuzzy neural network and FCMAC. Under the condition of a similar number of parameters, the type-2 FCMAC achieves the best performance with lowest computational cost. The type-2 FCMACs are more able to deal with the influence of system uncertainty and external disturbances. In this paper, current flight control law is adopted in the type-2 FCMAC controller design. The proposed controllers are implemented in a DSP. Tracking performance and adaptive capability are demonstrated through hardware simulations. Using PID, CMAC, type-1 FCMAC and type-2 FCMAC, the wind speed of turbulence limits are 30, 58, 90 and 100 ft/sec, respectively. The adaptive neural network controller and fuzzy neural network controller can overcome turbulence of up to 75 ft/s (Juang et al., 2011), and the recurrent neural network controller can overcome turbulence of up to 60 ft/sec (Juang et al., 2008). In this study, with optimal control gains, the CMAC control scheme can negotiate turbulence of up to 80 ft/sec, type-1 FCMAC can manage 105 ft/sec, and the type-2 FCMAC can manage 114 ft/sec. The proposed controllers have better performance than those previously developed. These intelligent controllers can be used to replace conventional controllers, and can act as an experienced pilot and guide aircraft to safe landings in severe wind turbulence environments.

REFERENCES

Adewuya, A. A. (1996). New methods in genetic search with real-valued chromosomes. M.S. Thesis, Dept. of Mechanical Engineering, Massachusetts Institute of Technology, 1996.

Albus, J. S. (1975). A New Approach to manipulator control: the cerebellar model articulation control (CMAC). ASME Journal of Dynamic Systems, Measurement, and Control 97, 220-227.

Arantes, J. S., M. S. Arantes, C. F. M. Toledo and B. C. Williams (2015). A multi-population genetic algorithm for UAV path re-planning under critical situation. Proc. IEEE International Conference on Tools with Artificial Intelligence. 486-493.

Asai, S., H. Onuma, T. Hata, Y. Miyazawa and T. Izumi (1997). Development of flight control system for automatic landing flight experiment. Mitsubishi Heavy Industries Technical Review 34(3), 125-128.

Boeing Publication (2000). Statistical Summary of commercial Jet Airplane Accidents. Worldwide Operations 1959-1999.

Chaturvedi, D. K., R. Chauhan and P. K. Kalra (2002). Application of generalized neural network for aircraft landing control system. Soft Computing 6, 441-118.

Cohen, C. E., H. S. Cobb, D. G. Lawrence, B. S. Pervan, J. D. Powell and B. W. Parkinson (1995). Automatic Landing of a 737 Using GNSS Integrity Beacons, Proc. ISPA.

DDC-I (1995). Advanced auto landing system from swiss federal aircraft factory. Real-Time Journal, Sprint.

Eshelman, L. J. and J. D. Schaffer (1993). Real-coded genetic algorithms and interval-schemata. Foundations of Genetic Algorithms 2, 187-202.

Heimes, F. O., W. S. Ford and W. Land (2001). Application of evolutionary computation to aircraft control law design. Proc. Congress on Evolutionary Computation 2, 1040-1046,.

Holland, J. H. (1975). Adaptation in natural and artificial systems. Ann Arbor, MI, University of Michigan Press.

Holland, J. H. (1962). Outline for a logical theory of adaptive systems. Journal of the Association for Computing Machinery 3, 297-314.

Iiguni, Y., H. Akiyoshi and N. Adachi (1998). An intelligent landing system based on human skill model. IEEE Transactions on Aerospace and Electronic Systems 34(3), 877-882.

Imae, j., S. Nakatani and J. Takahashi (1999). GP based flight control in the windshear. Proc. IEEE International Conference on Systems, Man, and Cybernetics 2, 650-653.

Ionita, S. and E. Sofron (2002). The fuzzy model for aircraft landing control. Proc. AFSS International Conference on Fuzzy Systems, 47-54.

Izadi, H., M. Pakmehr and N. Sadati (2003). Optimal neuro-controller in longitudinal autolanding of a commercial jet transport. Proc. IEEE International Conference on Control Applications, CD-000202, 1-6.

Jorgensen, C. C. and C. Schley (1991). A neural network baseline problem for control of aircraft flare and touchdown. Neural Networks for Control, 403-425.

Juang, J. G., L. H. Chien and F. Lin (2011). Automatic landing control system design using adaptive neural network and its hardware realization. IEEE Systems Journal 5(2), 266-277, 2011.

Juang, J. G. and K. C. Chin (2004). Intelligent landing control based on neural-fuzzy-GA hybrid system. Proc. IEEE International Joint Conference on Neural Networks 3, 1781-1786.

Juang, J. G., H. K. Chiou and L. H. Chien (2008). Analysis and comparison of aircraft landing control using recurrent neural networks and genetic algorithms approaches. Neurocomputing 71, 3224-3238.

Juang, J. G. and W. P. Lin (2008). Aircraft landing control based on CMAC and GA techniques. Proc. of IFAC WC 2008, 1730.

Kanury, S. and Y. D. Song (2006). Flight management of multiple aerial vehicles using genetic algorithms. Proc. of the Thirty-Eighth Southeastern Symposium on System Theory, 33-37.

Kaufmann, D. N. and B. D. McNally (1995). Flight Test Evaluation of the Stanford University and United Airlines Differential GPS Category III Auotmatic Landing System, NASA Technical Memorandum 110355, June 1995.

Liang, Q. and J. Mendel (2000). Interval type-2 fuzzy logic systems: theory and design. IEEE Transactions on Fuzzy Systems 8(5), 535-550.

Liu, Z., Y. Zhang and Y. Wang (2007). A type-2 fuzzy switching control system for biped robots. IEEE Transactions on Systems, Man, and Cybernetics-Part C 37(6), 1202-1213.

Michalewicz, Z. (1992). Genetic algorithm + data structure = evolution programs. New York, Springer-Verlag.

Nho, K. and R. K. Agarwal (2000). Automatic landing system design using fuzzy logic. Journal of Guidance, Control, and Dynamics 23, 298-304.

Visual Solutions, Inc. (2002). VisSim User's Guide-Version 5.0.

Wang, C. H., C. S. Cheng and T. T. Lee (2004). Dynamical optimal training for interval type-2 fuzzy neural network. IEEE Transactions on Systems, Man, and Cybernetics-Part C 34(3), 1462-1477.

Zhang, F. and Y. Wang (2013). Automatic landing of unmanned aerial vehicle using fuzzy control. Proc. IEEE International Conference on Information and Automation, 472- 477, 2013.