12-2013

Adaptive computer-generated forces for simulator-based training, Expert Systems with Applications

Teck-Hou TENG

Ah-hwee TAN
Singapore Management University, ahtan@smu.edu.sg

Loo-Nin TEOW

Expert Systems with Applications 40 (2013) 7341–7353

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

Adaptive computer-generated forces for simulator-based training

Teck-Hou Teng [a,*], Ah-Hwee Tan [a], Loo-Nin Teow [b]

[a] School of Computer Engineering, Nanyang Technological University, Singapore
[b] DSO National Laboratories, Singapore

ARTICLE INFO

Keywords:
Simulator-based training
Reinforcement learning
Self-organizing neural network

ABSTRACT

Simulator-based training is in constant pursuit of increasing level of realism. The transition from doctrine-driven computer-generated forces (CGF) to adaptive CGF represents one such effort. The use of doctrine-driven CGF is fraught with challenges such as modeling of complex expert knowledge and adapting to the trainees' progress in real time. Therefore, this paper reports on how the use of adaptive CGF can overcome these challenges. Using a self-organizing neural network to implement the adaptive CGF, air combat maneuvering strategies are learned incrementally and generalized in real time. The state space and action space are extracted from the same hierarchical doctrine used by the rule-based CGF. In addition, this hierarchical doctrine is used to bootstrap the self-organizing neural network to improve learning efficiency and reduce model complexity. Two case studies are conducted. The first case study shows how adaptive CGF can converge to the effective air combat maneuvers against rule-based CGF. The subsequent case study replaces the rule-based CGF with human pilots as the opponent to the adaptive CGF. The results from these two case studies show how positive outcome from learning against rule-based CGF can differ markedly from learning against human subjects for the same tasks. With a better understanding of the existing constraints, an adaptive CGF that performs well against rule-based CGF and human subjects can be designed.

1. Introduction

Continued reliance on manually crafted doctrine-driven computer-generated forces (CGFs) for simulator-based training (Farmer, van Rooij, Riemersma, Jorna, & Moraal, 2003) is an untenable and costly proposition (Bell & Waag, 1998) with diminishing return. However, recent advancements made to variants of machine learning techniques (Tan, 2004; Teng, 2012) should allow the doctrine-driven CGF to be adaptive to the varying competencies of the trainees in the near future.

The works reported in Teng, Tan, Tan, and Yeo (2012b) and Teng, Tan, Ong, and Lip (2012a) represent one such development of intelligent and adaptive CGF for simulator-based training. A commercial-grade simulation platform known as the CAE STRIVE™ CGF (CAE Inc., 2007) is used to model 1-v-1 air combat maneuvering (ACM) scenarios. A self-organizing neural network based on a fusion architecture for learning and cognition (FALCON) (Tan, 2004) is used by a Blue CGF to learn air combat maneuvers that can out-maneuver a Red CGF. Using reinforcement learning (RL) (Sutton & Barto, 1998), it improves on its choice of air combat maneuvers based on feedback on the effect of the previous choices.

The CGF–CGF experiments reported in Teng et al. (2012b) uses doctrine-driven CGF as the opponent to the adaptive CGF. Four sets of initial conditions are used in round robin to illustrate the ability to generalize and learn efficiently (Teng et al., 2012b). Subsequently, two blind CGF–Human experiments reported in Teng et al. (2012a) used human pilots as the opponent to the adaptive CGF. The adaptive CGF showed some amount of adaptation to score a temporary advantage over the trainee pilots in the 1st CGF–Human experiment. Consequentially, the trainee pilots correctly matched the desirable attributes to the adaptive CGF. Further validation at the 2nd CGF–Human experiment by the veteran pilots identify the need for more adaptive CGF.

This paper continues with the survey of the related works in Section 2. This is followed by a brief introduction of the self-organizing neural network in Section 3. More in-depth presentation on the use of the hierarchical doctrine are provided in Section 4. The air combat simulation platform used is introduced in Section 5. This is followed by the descriptions of the 1-v-1 ACM scenario in Section 6. Details on how the CGF–CGF experiments and the CGF–Human experiments are conducted are provided in Sections 7 and 8 respectively. This is followed by the comparisons of the quantitative results from the CGF–CGF and the CGF–Human experiments in Section 9. Direct comparisons of the qualitative assessments by the human pilots are presented in Section 10. Last but not least, Section 11 concludes this work and offers some implications and the suggested plans for this work.

* Corresponding author. Address: Blk N4-B1a-02, 50 Nanyang Avenue, Singapore 639798, Singapore. Tel.: +65 9271 9340.
E-mail address: thteng@ntu.edu.sg (T.-H. Teng).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.07.004

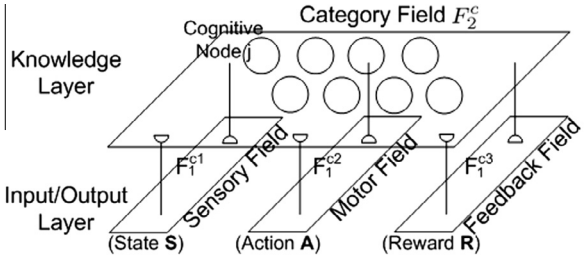

Fig. 1. The FALCON architecture.

2. Related work

Expert systems are known to be used for military applications such as advanced visual target recognition, autonomous tactical vehicles and combat pilot aid systems (Gilmore, 1985) and combat games (Ardema & Rajan, 1987). Earlier simulations of air combat applied the concept of differential game (Grimm & Well, 1991). Later on, artificial neural network (ANN) was being used for air combat (Rodin & Amin, 1992). Competition-based learning of evasive maneuvers of a plane from a missile was demonstrated using SAMUEL (Grefenstette, Ramsey, & Schultz, 1990). More recently, using computer-generated forces (CGF) as sparring partners has gained in popularity (Wray, Laird, Nuxoll, Stokes, & Kerfoot, 2005).

Separately, the Soar architecture was used to model complex knowledge in the 'live' air combat missions (Jones et al., 1999). However, it is still incapable of expanding its knowledge as available through this work. Subsequent extension of Soar with reinforcement learning was demonstrated using a different problem domain (Nason & Laird, 2005). The use of agent-oriented approach to model fighter combat was also reported (Heinze, Smith, & Cross, 1998). However, it was also reported that various challenges such as the lack of cognitive credibility, the capture of expert knowledge and other human factors remain unresolved. In another work, the Adaptive Neuro-Fuzzy Inference System was used to control the flight of unmanned air vehicles (UAVs) along the pre-defined trajectories with just some instability under certain flight conditions.

A two-layer hybrid multi-agent architecture known as ACOM-SIM was also used to model a cognitive agent that encompasses a symbolic world model and a reactive agent that reacts to environmental stimulations for asymmetric land combat (Cil & Mala, 2010). The development of mission-critical expert system reported in Bloom and Chung (2001) is similar to the developmental phase of this work. However, that work was conducted for the auto-planning of air battle whereas this work explores the use of self-organizing neural network to discover new knowledge with the only initial participation of subject matter experts (SMEs).

In addition, a Close Combat Tactical Trainer (CCTT) and the Training Exercise Development System (TREDS) to schedule manned simulator modules comprising of armored and mechanized forces (McGinnis & Phelan, 1996). Together, the complete system had improved unit readiness, resource utilization and save cost through better training management. Another expert system known as the Maneuver and Fire Support Planner (MFSP) was also developed to evaluate operational plans (Pereira, Sanchez, & Rives, 1999). An agent known as the Plan Viewer (PV) and another agent known as the Global Planner (GP) were created to allow commanders to manipulate military units and to generate automatic knowledge-based simulation of the operational plan respectively. BDI-based agent known as JACK was also used to create an automated wingman (Wallis, Ronnquist, Jarvis, & Lucas, 2002). The UAV is defined using JACK concepts such as goals, plans, world events and maintenance conditions. Though use of BDI-based approach was effective, there was still some difficulty in the formulation of tactical behaviors.

In another work, a fuzzy multi-criteria decision making method was first used to determine the importance weights of evaluation criteria (Wang & Chang, 2007). Using the synthesized ratings of candidate aircraft, TOPSIS was applied to obtain a crisp overall performance value of the candidate aircraft for making the final decision. Their proposed expert system helped them to identify the best-performing training aircraft. Separately, the Tactical Decision-Making Under Stress (TADMUS) (Smith, 2004) was developed to support recognition-primed decision making and explanation-based reasoning. It was evaluated to be better at identifying deceptive threats than the crew members. A more recent work applies Case-Based Reasoning and Bayesian Belief Networks (BBNs) to create a smart decision support system (SDSS). It is used for the assessment of critical success factors (CSFs) in military decision-making (Louvieris, Gregoriades, & Garn, 2010). The BBNs are used to quantify the relative strength of the CSFs.

Of similar objectives to this work, behaviors of the fighter pilots during air combat are mined using three behavior prediction patterns was proposed (Yin, Gong, & Han, 2011). Data structures are defined for each of these behavior prediction patterns. Subsequently, behaviors of the fighter pilots are mined in three phases using the behavior mining algorithm. Experimental results illustrated the effectiveness of the proposed fighters behaviors mining method.

3. The self-organizing neural network

Based on the adaptive resonance theory (ART) (Carpenter & Grossberg, 1987), the self-organizing neural network that derives from FALCON is capable of learning incrementally in real time. As a function approximator, it generalizes on the vector patterns without compromising on its prediction accuracy. Action policies are discovered through real-time interactions with the environment using reinforcement learning (Tan, Lu, & Dan, 2008). The value of applying the action choices on the states is estimated using a temporal difference method known as *Q*-Learning (Watkins & Dayan, 1992).

3.1. Structure and operating modes

Structurally, the FALCON network (Tan, 2004) has a two-layer architecture (see Fig. 1), comprising of an input/output (IO) layer and a knowledge layer. The IO layer has three input fields, namely a sensory field F_1^{c1} for accepting state vector **S**, an action field F_1^{c2} for accepting action vector **A**, and a reward field F_1^{c3} for accepting reward vector **R**. The category field F_2^c in the knowledge layer stores the committed and uncommitted cognitive nodes. Each cognitive node j has three fields of template weights $\mathbf{w}^{ck}$ for $k = 1,\ldots,3$.

FALCON has three modes of operation – INSERT, PERFORM and LEARN. It operates in the *PERFORM* mode to decide on action choices for the states. It operates in the *LEARN* mode to learn the effect of these action choices on the states. It operates in the *INSERT* mode to assimilate domain knowledge into itself (Teng & Tan, 2008; Teng, Tan, & Tan, 2008). The Fusion ART algorithm outlined in Algorithm 1 is used to find a winning cognitive node J to serve the purpose of these three modes of operation.

Algorithm 1. The fusion ART algorithm

Require Activity vectors $\mathbf{x}^{ck}$ and all weights vector $\mathbf{w}_j^{ck}$
1: **for** each F_2^c node j **do**
2: **Code Activation**: Derive choice function T_j^c using

$$T_j^c = \sum_{k=1}^{3} \gamma^{ck} \frac{\left|\mathbf{x}^{ck} \wedge \mathbf{w}_j^{ck}\right|}{\alpha^{ck} + \left|\mathbf{w}_j^{ck}\right|}$$

where the fuzzy AND operation $(\mathbf{p} \wedge \mathbf{q})_i \equiv min\ (p_i, q_i)$, the norm $\|.\|$ is defined by $|\mathbf{p}| \equiv \sum_i p_i$ for vectors $\mathbf{p}$ and $\mathbf{q}$, $\alpha^{ck} \in [0,1]$ is the choice parameters, $\gamma^{ck} \in [0,1]$ is the contribution parameters and $k = 1,\ldots,3$
3: **end for**
4: **repeat**
5: **Code Competition**: Index of winning cognitive node J is found using

$$J = \arg\max_j \left\{ T_j^c : \text{ for all } F_2^c \text{ node } j \right\}$$

6: **Template Matching**: Check whether the match functions m_j^{ck} of cognitive node J meet the vigilance criterion

$$m_j^{ck} = \frac{\|\mathbf{x}^{ck} \wedge \mathbf{w}_J^{ck}\|}{\|\mathbf{x}^{ck}\|} \geqslant \rho^{ck}$$

where $\rho^{ck} \in [0,1]$ for $k = 1,\ldots,3$ are the vigilance parameters
7: **if** vigilance criterion is satisfied **then**
8: *Resonance State* is attained
9: **else**
10: **Match Tracking**: Modify state vigilance ρ^{c1} using

$$\rho^{c1} = \min\left\{ m_J^{ck} + \psi, 1.0 \right\}$$

where ψ is a very small step increment to match function m_J^{ck}
11: **Reset**: $m_J^{ck} = 0.0$
12: **end if**
13: **until** *Resonance State* is attained
14: **if** operating in LEARN/INSERT mode
15: **Template Learning**: modify weight vector $\mathbf{w}_J^{ck}$ using

$$\mathbf{w}_J^{ck(\text{new})} = (1 - \beta^{ck})\mathbf{w}_J^{ck(\text{old})} + \beta^{ck}(\mathbf{x}^{ck} \wedge \mathbf{w}_J^{ck(\text{old})})$$

where $\beta^{ck} \in [0,1]$ is the learning rate
16: **else if** operating in PERFORM mode
17: **Activity Readout**: Read out the action vector $\mathbf{A}$ of cognitive node J using

$$\mathbf{x}^{c2(new)} = \mathbf{x}^{c2(old)} \wedge \mathbf{w}_J^{c2}$$

Decode $\mathbf{x}^{c2(new)}$ to derive recommended action choice a
18: **end if**

3.2. Incorporating temporal difference method

Outlined in Algorithm 2, a temporal difference (TD) method is used to estimate the value functions of state-action pairs $Q(s,a)$ (Tan et al., 2008). Using feedback from the environment on the performed action a selected using Algorithm 1, the Q-value $Q(s,a)$ is estimated using a TD formula. This estimated Q-value is used as the teaching signal to FALCON to learn the association of the current state s and the performed action a.

Algorithm 2. TD-FALCON algorithm

1: Initialize FALCON
2: Sense the environment and formulate a state representation s
3: Choose to explore at a probability of ϵ
4: **if** Exploration **then**
5: Use *Exploration Strategy* (Teng and Tan, 2012) to select an action choice a
6: **else if** Exploitation **then**
7: Use *Direct Code Access* (Tan, 2007) to select an action choice from existing knowledge
8: **end if**
9: Use action choice a on state s for state s'
10: Evaluate effect of action choice a to derive a reward r from the environment
11: Estimate the Q-value function $Q(s,a)$ following a temporal difference formula given by $\Delta Q(s,a) = \alpha TD_{err}$
12: Present **S**,**A** and **R** for **Learning**
13: Update the current state $s = s'$
14: Revise ϵ using Line 4–14 of Algorithm 3
15: Repeat from Step 2 until s is a terminal state

Iterative value estimation: The temporal difference method incorporated into FALCON is known as the Bounded Q-Learning (Tan et al., 2008). It estimates the value of applying action choice a to state s iteratively. The updated Q-value function $Q(s,a)^{(new)}$ is estimated using

$$Q(s,a)^{(new)} = Q(s,a)^{(old)} + \alpha TD_{err}(1 - Q(s,a)),$$

where $\alpha \in [0,1]$ is the learning parameter and the TD_{err} is the temporal error term which is derived using

$$TD_{err} = r + \gamma \max_{a'} Q(s',a') - Q(s,a),$$

where $\gamma \in [0,1]$ is the discount parameter and the $\max_{a'} Q(s',a')$ is the maximum estimated value of the next state s' and r is the immediate reward value. In this work, the immediate reward r is derived using

$$r = \frac{\sum_p^p \kappa_p \tau_p}{\sum_p^p \tau_p},$$

3.3. Self-regulating action exploration

The self-regulating action exploration approach outlined in Algorithm 3 (Teng, Tan, & Tan, 2012c) modifies the ϵ-Greedy method with linear ϵ-decay schedule to regulate exploration with respect to the learning process. Exploration is similarly performed with a probability of ϵ where $\epsilon \in [0,1]$. The difference is that an interval success rate ϕ derived using $\phi = \frac{w_s}{w_n}$ where w_s is the number of successful trials within w_n training iterations is used to revised ϵ as $1 - \phi$ after every $\mathcal{N}_w$ training iterations.

Algorithm 3. Self-RegulatingAction Exploration (SRE)

1: Initialize $\epsilon_0, \mathcal{N}_0, \mathcal{N}_\delta$
2: Initialize θ using

$$\theta = \frac{\epsilon_0}{\mathcal{N}_\delta \mathcal{N}_0}$$

where ϵ_0 is the initial value of ϵ, $\mathcal{N}_0$ is the initial number of training iterations and $\mathcal{N}_\delta$ determine the number of training iterations after $\epsilon = 0$

3: Set $\mathcal{N} = \mathcal{N}_0$
4: **for** $n = 0$ to $\mathcal{N}$ **do**
5: **if** $(n \bmod \mathcal{N}_w) \neq 0$ **then**
6: Linearly decay ϵ using θ
7: Tracks $\mathcal{N}_p$
8: **else if** $(n \bmod \mathcal{N}_w) \equiv 0$ **then**
9: Derive λ using $\frac{\mathcal{N}_p}{\mathcal{N}_w}$
10: Revise ϵ using

$$\epsilon^{\text{new}} = f(1-\lambda)\{\kappa(1-\lambda) + (1-\kappa)\epsilon^{\text{old}}\}$$

where $\kappa \in [0.0, 1.0]$ is the ϵ-adaptation rate and $f(x)$ is a step function such that

$$f(x) = \begin{cases} 1 & \text{when } x > 0 \\ 0 & \text{when } x \leqslant 0 \end{cases}$$

11: Derive $\mathcal{N}_r$ using

$$\mathcal{N}_r = \frac{\epsilon^{new}}{\mathcal{N}_\delta \theta} - \left(\frac{\epsilon^{new}}{\mathcal{N}_\delta \theta} \bmod \mathcal{N}_w\right) + \mathcal{N}_w$$

12: Update $\mathcal{N}$ using $\mathcal{N}_e + \mathcal{N}_r$ where $\mathcal{N}_e$ denotes the elapsed training iterations
13: Reset $\mathcal{N}_p$
14: **end if**
15: **end for**

Subsequently, ϵ is linearly decayed over the next $\mathcal{N}_w$ training iterations using an ϵ-decay rate δ. Such an approach gradually increases exploitation of the learned action policies within $\mathcal{N}_w$ training iterations. Using this approach, the effectiveness of the learned action policies to the states is continually evaluated.

4. Use of doctrine

In this work, doctrine refers to a collection of domain knowledge for performing specific task. Fig. 2 illustrates how the doctrine is used to extract the domain knowledge, the state space $\mathcal{S}$ and the action space $\mathcal{A}$. It is also indicated that the hierarchical doctrine will have to be flattened before it can be used. Details on how it can be flattened and used for the extraction of domain knowledge, state space $\mathcal{S}$ and action space $\mathcal{A}$ are already presented in Teng et al. (2012b).

4.1. Knowledge representation

In many real-world problem domains, it is inefficient to consider continuous-valued attributes by its values. Range-based consideration of attribute is seen as a more robust approach to handle continuous-valued attributes. Information on the possible bindings of the relevant attributes can be acquired from the existing doctrine or included under the advice of the SMEs.

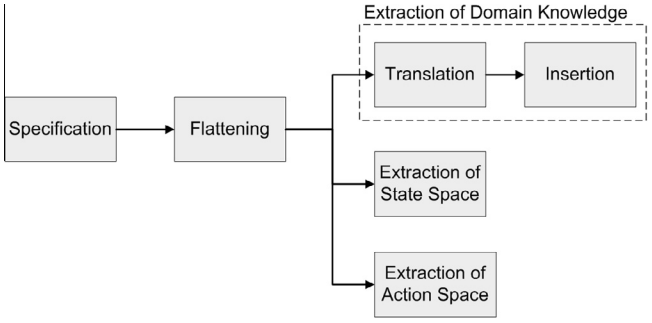

Fig. 2. Use of the doctrine for extraction of domain knowledge, state space and the action.

An attribute α in the antecedent of an IF-THEN rule is always bounded to either a value or a range. Each binding of attribute α is taken to be an atomic unit with Boolean outcome. It may be expressed using $\alpha - \theta - x$ where α is the attribute, θ represents the binding operator and $\theta \in \{=, \neq, \leqslant, <, >, \geqslant\}$ and x may either be a value or a range and $x \in \{\text{any possible values of attribute}\alpha\}$.

Using inequalities such as $<, \leqslant, >, \geqslant$ as the binding operator relaxes the binding of the attributes leading to a more generalized treatment of the attributes. For example, an expression such as $1000 < \text{range} < 2000$ is much more compact as compared to the direct encoding of 1001 possible values of the attribute range. Furthermore, the negation of an attribute binding can be expressed in a compact manner. For example, $\neg(1000 < \text{range})$ is actually equivalent to $1000 \geqslant \text{range}$.

4.2. The hierarchical structure

The domain knowledge whose knowledge representation scheme is presented in Section 4.1 has a hierarchical structure illustrated in Fig. 3. It has an inverted tree-like structure where the root node is at the topmost level. The final outcome of the decision-making process is found at the leaf nodes.

Each node in Fig. 3 contains several propositional rules arranged in similar tree-like structure. The consequent of higher level rules is used in the antecedent at lower level rules. The consequent of the lowest level rules of these nodes *points* to a lower level node of the outer rule hierarchy.

A small number of attribute–value bindings are considered at the antecedent. The consequent of the rules at the inner rule hierarchy has different types of attribute–value bindings. Such scattered definition of the domain knowledge is not suitable for use by FALCON. Therefore, a flattening procedure is included to provide

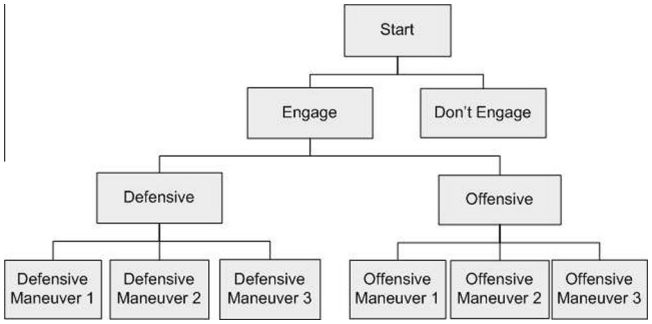

Fig. 3. A doctrine with a hierarchical structure.

an alternative representation of a semantically-equivalent set of domain knowledge.

4.3. The flattening procedure

A chain of rules is formed dynamically each time a decision is made. The idea is to substitute the consequent found in the antecedent of the lower level rules by the antecedent at the higher level rules. It becomes non-trivial when rules at the same level have to be broken up when there is disjunctive relation at the antecedent and have to simplified using *De Morgan's Law* before they are linked to rules at the adjacent levels. The Flattening procedure of the hierarchical doctrine is represented as Algorithm 4 (Teng et al., 2012b).

Algorithm 4. Flattening of Hierarchical Doctrine

1: Form chains of rule sets according to the hierarchical rule structure
2: **for** each rule set within each rule set chain **do**
3: **repeat**
4: Split Rules with *Disjunctive Relation*

$a \vee b \rightarrow c \equiv a \rightarrow c$ and $b \rightarrow c$

5: Apply *De Morgan's Law* on Antecedents

$\neg(a \vee b) \equiv \neg a \wedge \neg b$
$\neg(a \wedge b) \equiv \neg a \vee \neg b$

6: **until** antecedent of all rules has the following format

$a \wedge b \ldots \wedge z$

7: Link rules with *Antecedent-Consequent Dependencies*

$a \rightarrow b$ and $b \rightarrow c \equiv a \rightarrow c$

8: Repeat Line 5 for rules with antecedent of $\neg(a \wedge b)$ format
9: **end for**
10: **return** Flattened Doctrine

The Flattening procedure in Algorithm 4 satisfies three conditions required by FALCON to process propositional rules as patterns. The first condition is for rules that map the necessary conditions to the final action choices because the cognitive nodes in the category field F_2^c do not establish the dependency relationship among themselves. Due to the specific formulation of the choice function T_j^c and the match function $\mathbf{m}_j^{ck}$, the second condition is for the propositional symbols of the antecedent with only conjunctive relationship. Thirdly, antecedent with compounded knowledge structure such as $\neg(a \wedge b)$ or $\neg(a \vee b)$ has to be simplified using *De Morgan's Law* because FALCON is incapable of handling such compounded knowledge structure.

4.4. Extraction of state space and action space

Details on how the state space $\mathcal{S}$ and action space $\mathcal{A}$ are extracted from the flattened doctrine can be found in Section IV-B of Teng et al. (2012b). Outstanding issues on the extraction of the state space and action space are addressed separately in the following paragraphs.

State space: Using the proposed approach, the state space $\mathcal{S}$ is formed using only the propositional symbols. Each propositional symbol is a higher level representation of the raw sensory information sensed from the operating environment. Unlike the raw sensory information commonly represented using discrete or analog values, a propositional symbol can only be either asserted or unasserted. Consequentially, the number of propositional symbols for each attribute may vary in accordance to the complexity level of the tasks. However, only one of the propositional symbols of the attribute can be asserted at any one time.

This approach constrains the decision-making and learning tasks on the choice of attributes, the possible values of these attributes and the relationship between these attributes and the values. This means the resolution of $\mathcal{S}$ of the adaptive agent is constrained to match that of the non-adaptive agent. In this case, the strength of the adaptive agent is on the ability to learn states represented using unfamiliar combinations of the propositional symbols.

Action space: Using the proposed approach, the action choices of $\mathcal{A}$ are extracted from the consequent of the resultant propositional rules from the flattening process described in Section 4.3. Though it is possible to act on several aspects of the operating environment at each state, the scope of this work is limited to respond using just one type of action choice. This approach simplifies the tasks of discovering action policies during exploration and reduces the model complexity. However, it still turns out to be a fairly non-trivial task to identify the action choices effective to the states.

4.5. Encoding and insertion of domain knowledge

The flattened doctrine is comprised of single-layer propositional rules that can be used by FALCON as domain knowledge. Earlier works have demonstrated the insertion of domain knowledge for toy problem domains such as minefield navigation task (MNT) (Teng et al., 2008) and the Mission-On-Mars (MoM) problem domain (Teng & Tan, 2008). The insertion of professionally specified doctrine was also demonstrated for the ACM problem domain (Teng et al., 2012b).

The foci in these works are on the approaches used and the resultant effect of inserting domain knowledge into FALCON on the learning efficiency and the model complexity. Similar to those works, domain knowledge defined using propositional rules are translated into vector pattern for insertion into FALCON. The focus of the presentation is on the knowledge representation and the specific encoding scheme used.

5. The air combat simulation platform

Combination of a suite of commercial-grade simulation software and an own implementation of the Cognitive Engine are used in client–server configuration to simulate the 1-v-1 ACM scenarios. The Cognitive Engine is implemented as the client-side module while a suite of commercial-grade simulation software is used at the server side. Brief introduction to the components at the server side, the client side and the communication interface are provided here.

5.1. The server side

Collectively known as the CAE STRIVE™ CGF studio, the main components at the server side are the wire-frame display of the three-dimensional flight path (see Fig. 5), the flight canvas of the studio (see Fig. 4) and the entity properties panel (see Fig. 6). During simulation, the activities of the CGFs can be observed using the

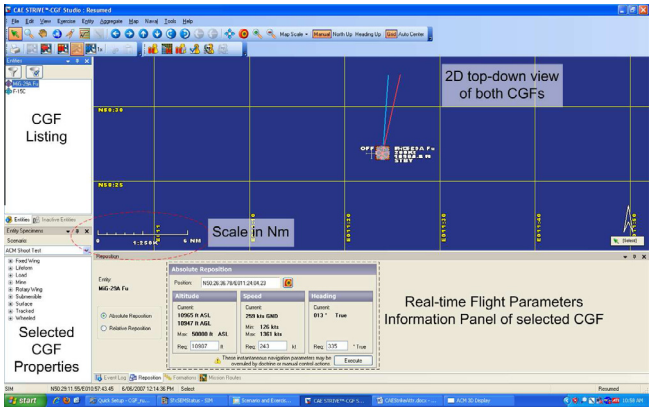

Fig. 4. The flight canvas of CAE STRIVE™ CGF Studio.

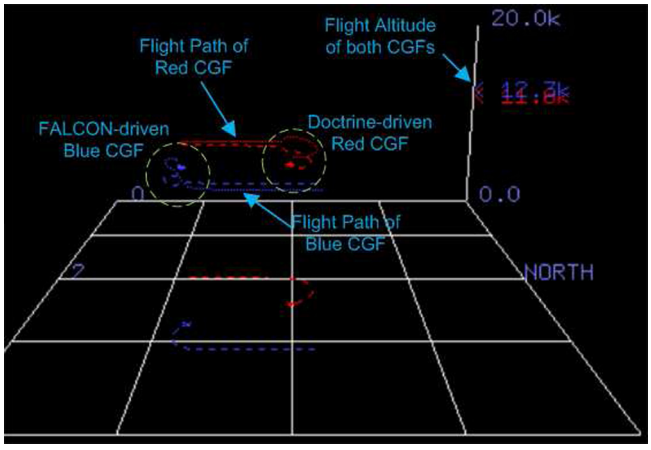

Fig. 5. Three dimensional view of the both CGFs.

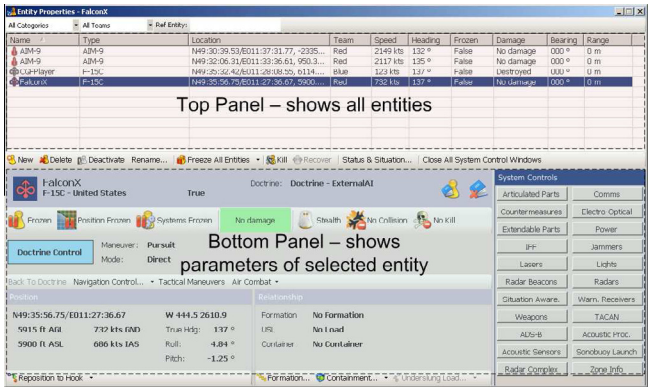

Fig. 6. The entity properties of CAE STRIVE™ CGF studio.

flight canvas. Real-time flight parameters of the selected CGF include the absolute position expressed as GPS coordinates, the altitude, the air speed and the heading during flight can also be monitored using the studio. These parameters are used to configure the initial conditions of the CGFs prior to flight.

The three-dimensional ACM display (3D display) shown in Fig. 5 illustrates the altitudinal differences and the absolute positions in three-dimensional airspace of the CGFs. The flight trajectories of the CGFs drawn using colored dotted lines are cleared at fixed time interval. The 3D display can also be rotated and scaled to follow the flight trajectory with greater clarity.

Seen in Fig. 6, more detailed information on the CGF are available through the entity properties panel (properties panel). The top panel shows all the entities together with a salient collection of real-time parameters. As observed, missiles and flares launched by the CGFs are also shown in the top panel. The bottom panel of the properties panel provides access to all the parameters of the selected entities. It includes information on the flight maneuver the CGF is currently executing, the doctrine it is attached with, a large collection of flight controls and also the designated role of the selected CGF in a team (if formed).

5.2. The client side

An own implementation of a Cognitive Engine known as DSOCA-STRIVE™ is used as the client. It includes the model layer, communication layer and the simulation layer. The *model layer* is an implementation of FALCON as an entity agent. This agent entity implements the basic sensory, actuation and decision-making mechanisms. The sensory and actuation mechanisms bring information into and out of FALCON respectively. The decision-making mechanism is used to decide on air combat maneuver, firing of weapons and the launching of flares at different time interval. The *simulation layer* includes the data logging and dissemination mechanism and the mechanism to implement iterative execution of the simulation. The *communication layer* facilitates the interaction with the human users and also with the server.

The entire simulation routine is controlled using the graphical user interface (GUI) of DSOCA-STRIVE™ illustrated in Fig. 7. Main panels of the GUI include the feedback panel, the server control panel, mission control panel, the CGF information panel and the simulation control panel. The mission control panel affects a particular training iteration (mission) while the controls at the simulation control panel dictate the entire simulation. Other than observing the dogfights between the CGFs using the flight canvas of the CAE STRIVE™ CGF studio manager at the server side, real-time feedback on the status of the simulation process and also from own CGF are also available at the Feedback panel seen in Fig. 8.

5.3. Client–server communication

The client–server configuration is set-up as illustrated in Fig. 9 using commercial-grade High Level Architecture interface known as CAE ICON™. In this work, the server and client are run using separate terminals linked using cross-linked Ethernet cable. Parts of the CAE ICON™ interface are installed at the server and client to implement the communication interface. From Fig. 9, the server forwards three types of information to the client. The server

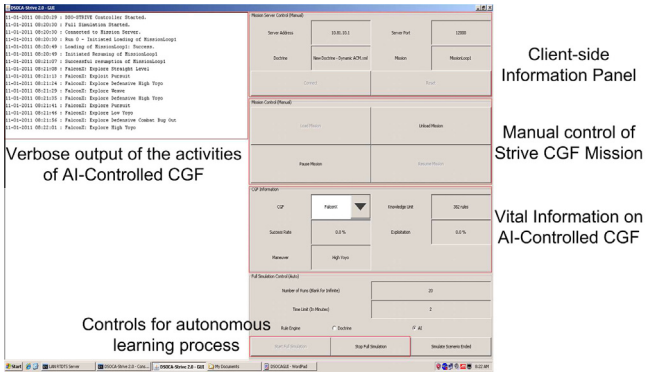

Fig. 7. The client-side Graphical User Interface of DSOCA-STRIVE™.

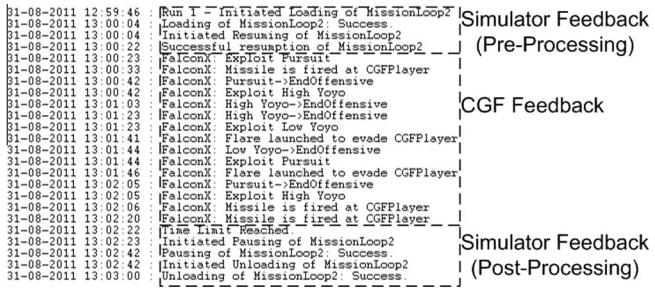

Fig. 8. The feedback panel of DSOCA-STRIVE™.

forwarded information such as the CGF's own parameters (EntityParameters), parameters of the flight maneuver (EntityACMParameters) and the relative parameters with respect to the other CGF (RelativeParameters) through the CAE ICON™ interface. On the other hand, the client forwards information such as the selected air combat maneuver, the weapon to be fired and the flare to be launched through CAE ICON™ to the server.

6. 1-v-1 air combat scenario

Based on the classical 1-v-1 pursuit-evasion problem in three-dimensional airspace (Ardema & Rajan, 1987), the scenario is implemented using the simulation platform presented in Section 5. Unlike (Grefenstette et al., 1990) where tactical plans are learned, learning of air combat maneuvers is conducted in real time using reinforcement learning. As illustrated in Fig. 10, a Red CGF and a Blue CGF are tasked to out-maneuver each other in a dogfight.

Similar scenarios are used for the CGF–CGF experiments reported in Teng et al. (2012b) and the CGF–Human experiments reported in Teng et al. (2012a). The similarities include using the same type of aircraft model for the CGFs and the human pilots. Both aircrafts are initialized to begin flying at the same altitude and the same air speed. All CGFs select their choice of air combat maneuvers using the rule-based knowledge from the same doctrine. The CGFs and the human pilots are able to engage each other using the same number of missiles and flares. The same set of constraints for the launching of missiles and flares are maintained for the CGFs and the human pilots.

6.1. The state space

For this work, a total of 15 simple and composite attributes are extracted from the ACM doctrine to form the state space $\mathcal{S}$. A composite attribute is made up of two or more simple attributes such as the composite attribute (mVtt–mVt) is made up of mVtt and mVt simple attributes. With reference to Fig. 11, the extracted state space includes relative parameters such as range and angular position, own entity ACM parameters such as current maneuver,

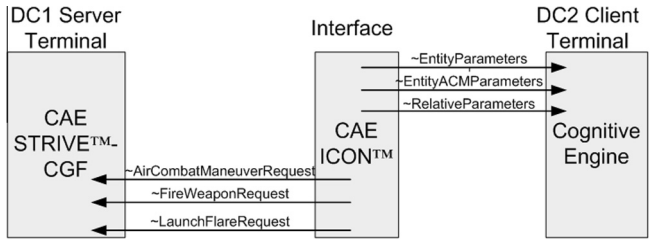

Fig. 9. An illustration of the client–server set-up between the AI terminal and Simulator terminal.

Fig. 10. Illustration of own CGF (Blue CGF) firing missile at the opponent (Red CGF). (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this article.)

maneuver lock status and own entity parameters such altitude and air speed. These are specified in the form of propositional symbols such as those seen in Table 1.

6.2. The action space

A set of 13 air combat maneuvers are used to form the action space $\mathcal{A}$. Specifically, there is one neutral maneuver, three offensive maneuvers and nine defensive maneuvers. The adaptive CGF learns to select air combat maneuvers effective to the situations. The execution of these air combat maneuvers is pre-programmed. The firing of missile and the launching of flare are not included as part of the action space because they are implemented as doctrine-driven behaviors.

6.3. Evaluative feedback

Use of feedback on the effect of action choices to the states is a signature approach in reinforcement learning. Selected reward attributes are monitored to provide intermediate reward r_i at the intermediate states and terminal reward r_t at the terminal states. Some details on how the intermediate reward r_i and the terminal reward r_t are formulated are provided in this section.

Intermediate reward: The intermediate reward r_i communicates the effect of the action choices to the intermediate states. Selected reward attributes such as the CGF's proximity to the weapon

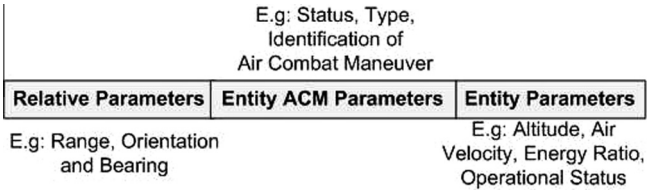

Fig. 11. The extracted state space.

Table 1
Samples of the propositional symbols in the state space.

No.	Type	Sample
1	Boolean	wOk = TRUE
2	Float (>)	$Abs(mTA) > 135°$
3	Float (<)	$mVtt - mVt < 50$
4	Float ($\leqslant$)	$mEnergyRatio \leqslant 0.85$
5	Float ($\geqslant$)	$Altitude \geqslant 4500$
6	String ($\neq$)	$mManeuver \neq Weave$

engagement zone (WEZ) of its opponent (dWEZ), the energy ratio of own CGF to its opponent (eRatio) and the threat level to own CGF with respect to its opponent (tLevel).

With reference to Fig. 12, the WEZ is an area around the opponent where air-to-air weapon can be fired with high hit probability of own CGF. Therefore, it is calculated using the proximity to opponent, target aspect (TA) angle and the antenna train angle (ATA) illustrated in Fig. 14 to ensure own CGF is outside of it. The ATA is the angular direction of the position of opponent with respect to current heading of own CGF whereas TA is the angular direction of the position of self with respect to the current heading of the opponent.

From Fig. 13, own CGF is less vulnerable when $\pm 0^\circ \leqslant \text{ATA} \leqslant \pm 90^\circ$ and more vulnerable when $\pm 90^\circ < \text{ATA} \leqslant \pm 180^\circ$. Therefore, own CGF need to select action choice at time n leading to ATA $(n) < \text{ATA}\ (n-1)$, TA $(n) > \text{TA}\ (n-1)$,ATA $(n) \to 0^\circ$ and TA $(n) \to 180^\circ$.

The energy ratio eRatio is derived using the energy level of both CGF derived using $H + \frac{V^2}{2G}$ where H is the altitude (in feet), V is the air speed and G is the acceleration due to gravity (32.2 ft/s^3). The threat level tLevel is derived using a set of heuristics to determine how threaten own CGF is by the opponent.

After deriving dWEZ,eRatio and tLevel, the intermediate r_i is subsequently derived using

$$r_i = \omega_{wez}\text{dWEZ} + \omega_{ratio}\text{eRatio} + \omega_{threat}\text{tLevel}$$

where ω_{wez},ω_{ratio} and ω_{threat} are the weights of the respective reward attributes and the intermediate reward $r_i \in [0.0, 1.0]$.

Terminal reward: The terminal reward r_t communicates the observed effect at the terminal states to the learning model. The terminal reward function is dependent on the number of observable outcomes at the terminal state. In this case, the three observable terminal outcomes and the terminal rewards are presented in Table 2.

The allocated value is aimed at giving a sense of desirability to each of these terminal outcomes. Therefore, from Table 2, the elimination of the opposing CGF (*HasKill*) is the best outcome while the elimination of own CGF (*IsKill*) is the worst outcome at the terminal state. A neutral outcome considered to be better than the worst outcome is assigned a value between the best and the worst outcome. It is used to estimate the value of choosing an air combat maneuver at the terminal states using the Bounded *Q*-Learning method.

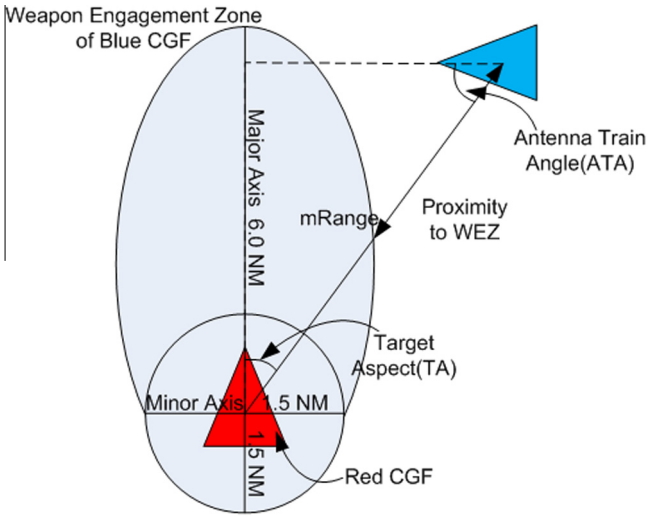

Fig. 12. An illustration of the Weapons Engagement Zone (WEZ) of the Adversary.

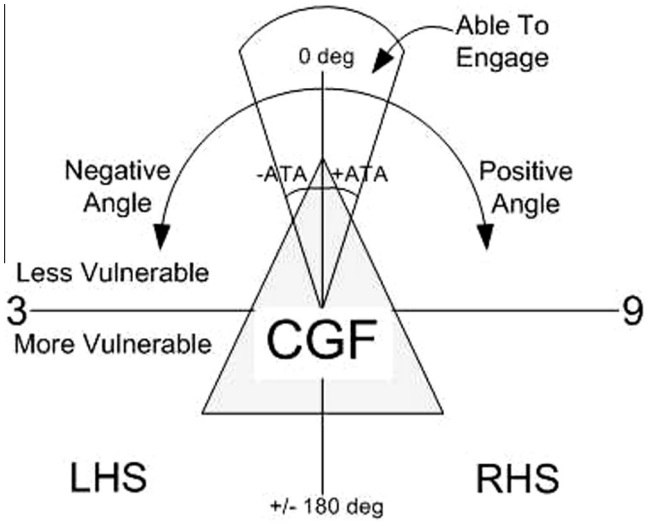

Fig. 13. An illustration of the vulnerability of CGF.

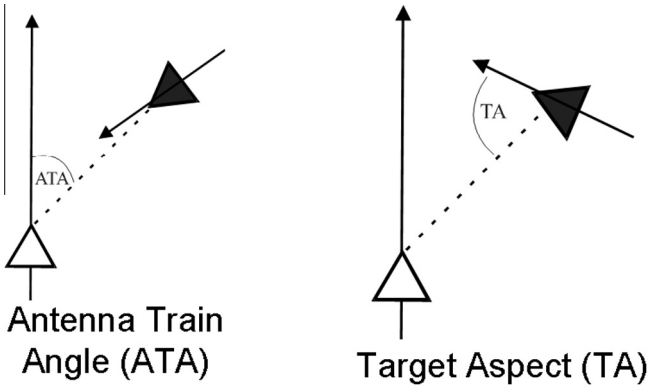

Fig. 14. Illustrations of the Antenna Train Angle (ATA) (left) and the Target Aspect (TA) (right) angular information.

Table 2
Terminal reward for 1-v-1 air combat maneuver.

No.	Outcome	Terminal reward
1	Eliminated Red CGF	1.0
2	Time-out of training iteration	0.5
3	Killed by Red CGF	0.0

7. The CGF–CGF ACM Experiments

The CGF–CGF ACM experiments are conducted using the simulation platform described in Section 5. The purpose of the experiments is to investigate the autonomous learning of air combat maneuvers during 1-v-1 dogfights in real time. For the experiments, the hierarchical doctrine used to drive the non-adaptive CGF is inserted into FALCON as the domain knowledge.

Four different initial positions illustrated in Fig. 15 are used in a round robin fashion in the experiments. This is to illustrate how FALCON is able to generalize learning to more than one initial conditions within the same session of reinforcement learning. The experimental results are collected from a session of reinforcement learning using these four sets of initial conditions. Each plotted data point in the plots is an average of 10 consecutive raw data points. This gives a total of 12 data points from a reinforcement learning process with 120 training iterations. The parameters used

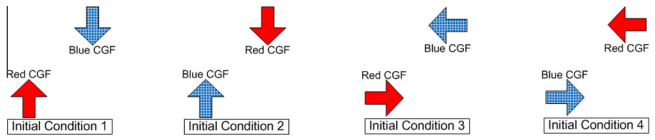

Fig. 15. An illustration of the four initial conditions used for the CGF–CGF experiments.

Table 3
Performance measures for air combat maneuver.

No.	Terminal State	Descriptions
1	HasKill	Blue CGF eliminates Red CGF
2	IsKill	Blue CGF is eliminated by Red CGF
3	EqualMatch	Blue and Red CGF survive the entire training iteration

for the CGF–CGF ACM experiments can be found in Teng et al. (2012b).

7.1. Performance measures

The performance measures for the air combat maneuver is closely tied to the terminal states. Specifically, the performance measures in Table 3 records the number of occurrences of each types of the terminal state during the simulation.

A favorable convergence of learning process is characterized by high percentage of *HasKill* which implies low percentage of *IsKill* and *EqualMatch*. However, in states where elimination of the opponent are difficult to achieve, high percentage of *EqualMatch* will be the next best outcome. For the experiments, the 1-v-1 air combats time-out in two minutes.

8. The CGF–Human ACM experiment

The CGF–Human ACM experiment was conducted to determine how the adaptive CGF perform against human pilots as its opponent. It is clear that such an arrangement is bound to increase the difficulty of learning air combat maneuvers against the Red CGF by several orders of magnitude. However, it had to be conducted to complete the study on the use of adaptive CGF for simulator-based training.

8.1. The human pilots

Two groups of pilots from different air force organisations are invited for the CGF–Human experiments. The first group of pilots comprises of two trainee pilots and the second group of pilots comprises of three veteran pilots. Such an arrangement allows a broader assessment of the CGFs.

Trainee pilots: The first group of participants are trainee pilots in their 20s. They had completed their basic military training stint and were undergoing vocational flight training on specific aircraft type. They had no *prior* experience using the commercial-grade simulation platform used in this work. However, almost all of them have some *flying* experience in some game-like flight simulators. Studies have shown that this can be translated to improved performance when it comes to actual flight (Gopher, Well, & Bareket, 1994).

Veteran pilots: The second group of participants are veteran combat pilots who had served their country in actual air combat missions as fighter jet pilots. Retired from regular services, they are engaged as the SMEs to CAE™ Inc. Being the consultants to the simulator products used in the CGF–Human experiment, they are much more familiar with the handling of the simulated aircraft than the trainee pilots.

8.2. Design of CGF–Human experiment

Though similar simulation configuration of the CGF–CGF experiment (Teng et al., 2012b) is used for the CGF–Human experiment, the participation of human pilots (Pausch, Crea, & Conway, 1992) demands closer attention on the timing parameters involved in the execution of the CGF–Human experiment. Some amount of time is allocated as intermission to rest the human pilots while they provide qualitative assessments of the CGFs they had flown against.

Each training iteration during reinforcement learning is referred to as a sortie and T_{sortie} refers to the duration of each sortie. A trial is the consecutive execution of multiple sorties and T_{trial} refers to the duration of each trial. A session is comprised of multiple trials and $T_{session}$ refers to the duration of each session. A cycle is a complete execution of the session by each human pilot. The number of cycles that can be completed in a day is denoted using N_{cycle}.

There are two groups of pilots and CGFs each. The first group of human pilots denoted using $\mathbf{P}_{nf}$ refers to the set of pilots who has not flown while the other group of human pilot denoted using $\mathbf{P}_f$ refers to the set of pilots who has flown. The first group of CGF denoted suing $\mathbf{C}_{nf}(P)$ refers to the set of CGF that has not flown against pilot P while the other group of CGF denoted using $\mathbf{C}_f(P)$ refers to the set of CGF that has flown against pilot P. After each session, $\mathbf{C}_f(P) \equiv \mathbf{C}$ while $\mathbf{C}_{nf}(P) \equiv \emptyset$ is emptied. The next session commences with $\mathbf{C}_{nf}(P) \equiv \mathbf{C}$. After each cycle, $\mathbf{P}_f \equiv \mathbf{P}$ and $\mathbf{P}_{nf} \equiv \emptyset$.

Several parameters need to be specified for the design of the CGF–Human experiment. These parameters include the estimated duration of each sortie T_{sortie}, the estimated duration of continuous flying before fatigue T_{alert}, the intermission between each trial T_{TR} and each session T_{SR} and the agreed amount of flying time (in minutes) in a day by the pilots T_{day}. From our perspective, the main concern is on the number of training iterations required before positive effect of learning can be observed. This is the desired number of sorties $N'_{sorties}(CGF)$ for each CGF. On the other hand, in view of their professional appointments, the pilots are concerned with the amount of time they need to commit to the CGF–Human experiment. Therefore, the calculated number of days T_{C-H} for the CGF–Human experiment have to be agreed before the experiment can proceed.

As an illustration using a sample schedule shown in Table 4, a 3-h orientation session is included for the pilots. The actual CGF–Human experiment may then commence after the lunch break. At Session 1, pilot HP2 is tasked to fly against S-CGF while pilot HP1 participates in the trial as a co-pilot. After a trial of N_{sortie} sorties and T_{TR} minutes of intermission, pilot HP2 continues in the next trial with L-CGF. A session of CGF–Human experiment is completed after pilot HP2 has flown against S-CGF and L-CGF. Session 2 begins after T_{SR} minutes of intermission with pilot HP1 as the pilot and pilot HP2 as the co-pilot. The actual choice of CGF is not known to the pilot during the trials.

Table 4
Sample schedule of CGF–Human experiment in a day.

Session	Time	Trial	CGF	Sortie	Pilot/Co-Pilot
0	0900–1030 h	Orient	S-CGF	30	HP1/HP2
	1035–1200 h	Orient	S-CGF	30	HP2/HP1
1	1400–1600 h	Actual	S-CGF	15	HP2/HP1
		Actual	L-CGF	15	HP2/HP1
2	1615–1815 h	Actual	L-CGF	15	HP1/HP2
		Actual	S-CGF	15	HP1/HP2

8.3. The questionnaires

The quantitative results from the CGF–Human experiment reveals only the final outcome of the sortie. Qualitative assessments are needed to capture aspects of the CGFs missed by the quantitative results. Therefore, two sets of questionnaires are used to gather qualitative assessments of the CGFs from the pilots after each trial and session. The pilots are briefed on the approximate definition of the attributes and the different frequencies of occurrence to ensure their qualitative assessments of the CGFs are conducted using a similar level of understanding of the terms. Specifically, the pilots were requested to assess the CGF on the following attributes.

Predictable	: CGF acts in a way that I can see a pattern to predict the next move.
Intelligent	: CGF is able to outsmart me with clever moves and plans.
Skillful	: CGF has good technique and skill in executing the air combat tactics.
Challenging	: CGF behaves in a way that stretches my resources to the limit.
Adaptive	: CGF is able to adjust and adapt to my actions automatically.
Aggressive	: CGF is tending or disposed to attack others.

Trial questionnaire: After each trial, the pilots are required to complete a *trial questionnaire* comprising of the first five attributes. Given that the requested attributes are qualitative in nature, the pilots are requested to conduct their assessments using the following perceived frequencies of occurrence: *never*, *rarely*, *sometimes*, *frequently*, *most of the time* and *always*.

Session questionnaire: After each session, the pilots are also required to complete a session questionnaire. Unlike the trial questionnaire, the pilot ranks all the CGF types he had flown against during that session using all six attributes. A CGF is ranked more highly when it is showing more of the specific attributes in comparison to the other CGF. To avoid biasing their qualitative assessments, the pilots are unaware of the actual identity of the CGFs when they rank the CGFs.

8.4. Conducting the CGF–Human experiment

The CGF–Human experiments were conducted using the initial conditions seen in Fig. 16. The physical set-up and the execution procedures of the CGF–Human experiment are described here. The specific handling of the CGFs during the CGF–Human experiment is also included as part of the execution procedures.

Physical set-up: The CGF–Human experiment is conducted using the flight training simulator set-up illustrated in Fig. 17. The physical set-up comprises of one unit of desktop workstation WS1, two units of desktop computer DC1 and DC2 and a pair of joysticks to simulate the HOTAS in the cockpit of a fighter jet. The joysticks are connected to the desktop computer DC1 running the commercial-grade simulator software known as CAE STRIVE™ CGF Studio.

From Fig. 17, DC1 is connected to the desktop computer DC2 running the cognitive engine in a client–server configuration illustrated in Fig. 9. The CAE ICON™ interface facilitates the communication between DC1 and DC2. DC1 is also connected to the desktop workstation WS1 designated as the Image Generator (IG) for generating the Out-of-The-Window (OTW) view used by the pilots. It can generate a multi-channel three-dimensional panoramic OTW view of 100° horizontal and 56° vertical of the airspace. A heads-up display (HUD) unit is superimposed onto the OTW view to provide an artificial horizon, the altitude, the rate of climb, the air speed, the G-meter and the compass heading of the aircraft. The weapons aiming system is omitted from the HUD.

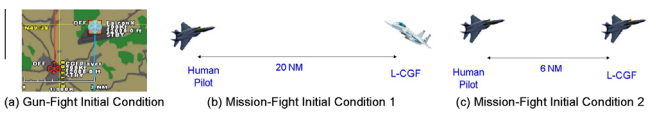

Fig. 16. Initial conditions used for the CGF–Human experiments involving the trainee pilots (a) and the veteran pilots (b and c).

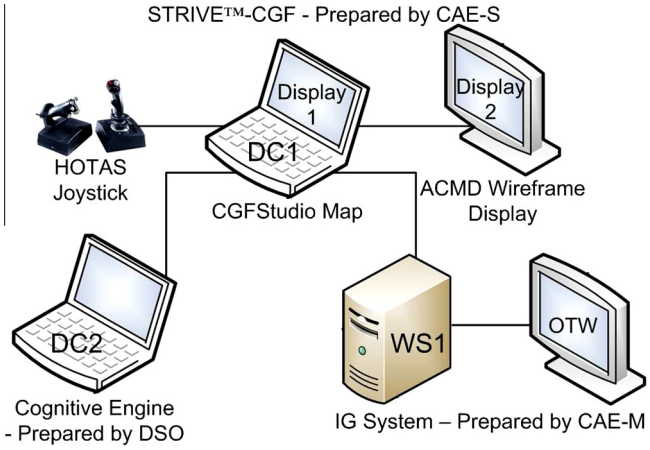

Fig. 17. The physical set-up of the CGF–Human experiment. It implements the logical set-up illustrated in Fig. 9.

Execution procedures: A team comprising of the pilot, the co-pilot and the support crews is necessary for the CGF–Human experiment. The pilot does the actual flying of the simulated aircraft to engage a CGF in 1-v-1 dogfights. Using the ACMD wireframe display and the CGFStudio Map in DC1, the *co-pilot* provides additional information such as the elevation and general position of the opponent when it is out of the OTW view. In addition, the *co-pilot* also records the number of missiles fired and the final outcome of the sorties as the quantitative results of the CGF–Human experiments.

The support crews comprising of the AI team and the simulator team ensure the smooth execution of the CGF–Human experiment. The simulator team is needed to make speedy recovery of the CGF–Human experiment when there is any technical glitch. The pilot flies against either the L-CGF or the S-CGF during a trial. On top of not revealing the identity of the CGF to the pilots, the order of appearance of the CGFs during the session is also randomized. The AI team ensures the correct configurations are used for the trials such that the same amount of experimental data is collected from all the pilots with all the CGFs.

Both CGFs are inserted with the same doctrine prior to the experiments. Only the L-CGF updates its knowledge base by learning its interactions with the human pilots. The AI team is required to ensure the updated knowledge base is used for the subsequent trials involving the L-CGF with any of the pilots. The AI team is also required to ensure the correct set of initial conditions is used for the trials when multiple sets of initial conditions are used.

The experiments: Two independent CGF–Human experiments were conducted using the Trainee and the Veteran pilots. The purpose of the experiments is to study, in comparison to a non-adaptive doctrine-driven CGF, how well the adaptive CGF can adapt against the pilots in 1-v-1 air combat scenario and to also gather qualitative assessments from the pilots on these two types of CGF. Further details on the CGF–Human experiment can be found in Teng et al. (2012a).

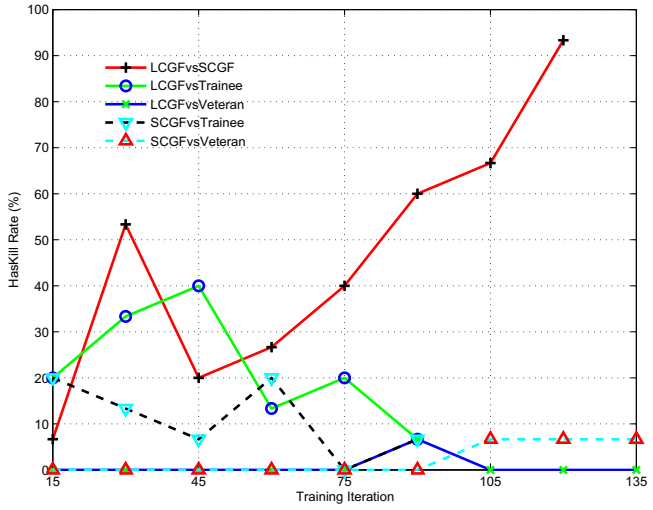

Fig. 18. Comparisons of the HasKill rates between the CGF–CGF and CGF–Human experiments.

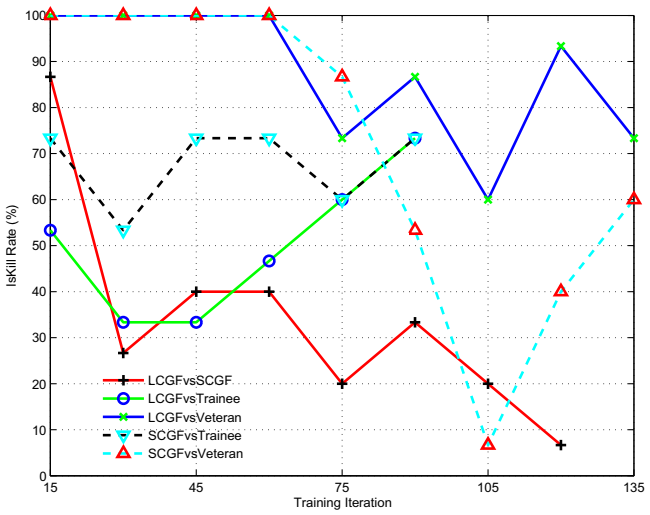

Fig. 19. Comparisons of the IsKill rates between the CGF–CGF and CGF–Human experiments.

9. Comparisons between CGF–CGF and CGF–Human experiments

The presentation on the CGF–CGF experiments in Teng et al. (2012b) and the CGF–Human experiments in Teng et al. (2012a) are on how the respective CGF performs with respect to the specific type of opponent. In contrast, the quantitative and qualitative results from these two types of experiments are compared directly here. Specifically, three sets of direct comparisons of the quantitative results from the CGF–CGF and the CGF–Human experiments are presented here. The CGF–CGF experiments (LCGFvsSCGF) were conducted between L-CGF and S-CGF while the CGF–Human experiments were conducted by flying L-CGF and S-CGF against the Trainee and Veterans separately. Changes to the initial conditions and the number of training iterations were included to adapt to the changing conditions at each experiments.

Results: The HasKill rates of these five configurations are compared in Fig. 18. The L-CGF in the LCGFvsSCGF configuration is seen improving on its HasKill rates over S-CGF to around 93% level from the 45th training iteration. The LCGFvsTrainee and SCGFvsTrainee configurations are seen having similar level of HasKill rates after 15 training iterations. However, L-CGF is able to improve on its HasKill rates over Trainee for the next 30 training iteration. In comparison, the HasKill rates of SCGFvsTrainee drops by more than 10% over the same duration. As the experiments progress, the HasKill rates of LCGFvsTrainee and SCGFvsTrainee configurations are seen falling. Notably, the HasKill rates of L-CGF over Trainee fall more slowly than that of S-CGF. In sharp contrast, the HasKill rates of L-CGF over the Veteran remain at the 0% level for almost the entire duration of the LCGFvsVeteran configuration. Similarly, the HasKill rates of S-CGF over the Veteran remain at 0% level until the 90th training iteration. Subsequently, it remain unchanged at 6.67% for the remaining training iterations.

The IsKill rates of these five configurations are compared in Fig. 19. It can be seen that the IsKill rates of L-CGF in LCGFvsSCGF falls quite consistently from around 88% to about 7% over 120 training iterations. The IsKill rates of L-CGF in LCGFvsTrainee have also fallen slightly before raising from 45th training iteration. In contrast, the fall in the IsKill rate of L-CGF in LCGFvsVeteran is only observed after 75 training iterations. It then fluctuates around the 70% level for the remaining training iterations. The fluctuation of IsKill rates of S-CGF in SCGFvsTrainee is less varied than that of S-CGF in SCGFvsVeteran. In contrast, a significant drop in IsKill rates of S-CGF in SCGFvsVeteran is observed at the 105th training iteration.

From the comparison of EqualMatch rates of these five configuration in Fig. 20, the EqualMatch rates of S-CGF in SCGFvsVeteran spikes at the 105th training iteration. In contrast, the EqualMatch rates of S-CGF in SCGFvsTrainee fluctuates below the 40% level. Notably, the EqualMatch rates of L-CGF with Veteran raise from 0% only after 60 training iterations. In comparison, the EqualMatch of L-CGF in LCGFvsTrainee stays above 20% for a large part of the training process. Unlike all the other configurations, the EqualMatch of L-CGF in LCGFvsSCGF raises to around 40% before falling to 0% after 120 training iterations.

Discussions: The complementary performance measures of HasKill, IsKill and EqualMatch presented in Section 7.1 are necessary to analyze the performance of the CGFs. The objective of these analyses is to show how differently L-CGF can adapt against the S-CGF and against the human pilots comparing to S-CGF. For the LCGFvsSCGF configuration, dip in the HasKill rates of L-CGF at the 45th training iteration means that L-CGF eliminates S-CGF less frequently. However, the gradual raise in the EqualMatch rates and the gradual drop in the IsKill rates over the same duration also mean that L-CGF has acquired knowledge that allow it to be more

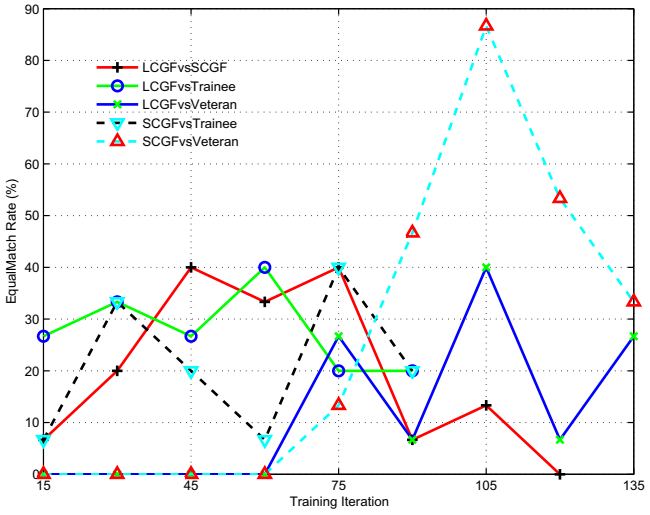

Fig. 20. Comparisons of the EqualMatch rates between the CGF–CGF and CGF–Human experiments.

challenging to S-CGF. This trend of positive learning continues to allow L-CGF to score about 93% HasKill rate after 120 training iterations.

Similarly, the HasKill rates of L-CGF in LCGFvsTrainee rise gradually over the first 45 training iterations. The gradual decline in HasKill rates is matched by a sustained EqualMatch rate for the next 45 training iterations. However, given that the Trainees adapted much more efficiency than L-CGF, the IsKill of L-CGF continues to increase as training progresses. Quite similarly, L-CGF in LCGFvsVeteran configuration is much less able to eliminate the Veteran. At best, it is able to learn positively enough to score some amount of EqualMatch against the Veterans.

10. Comparisons of qualitative asssessments between trainee and veteran pilots

Blind qualitative assessments of the CGFs under different settings are made by the human pilots after they have flown against it using the format presented in Section 8.3. For the qualitative assessment to be blind, the identity of the CGFs is withheld from the pilots. The qualitative assessment in Figs. 21 and 22 compares the qualitative assessments of L-CGF and S-CGF by the trainee pilots and the veteran pilots after each trial and each session.

Results: With reference to Fig. 21, the higher qualitative score means more of the quality the CGF is assessed to have exhibited during the trial. Therefore from Fig. 21, L-CGF is assessed by the Trainees to be the least predictable, the most intelligent, skillful, challenging and adaptive. In contrast, L-CGF is assessed by the Veterans to be the most predictable, the least intelligent and skillful. L-CGF is assessed by the Veterans to be of the same level of challenge as the Trainees find their S-CGF. However, the Veterans did find L-CGF more adaptive than the S-CGF they had flown against. In comparison, the Veterans assess S-CGF to be more predictable, less intelligent, skillful and adaptive than the Trainees' assessment of S-CGF on those qualities. In contrast, the Trainees found their S-CGF less challenging than the Veterans find their S-CGF.

The qualitative assessments of the CGFs in Fig. 22 may be interpreted in a similar way as those in Fig. 21. From Fig. 22, L-CGF was assessed by the Trainees to be more predictable than S-CGF they have flown against in the same session. This is in contradiction to their assessment of L-CGF on the predictable quality using the trial questionnaire seen in Fig. 21. Apart from that, the assessments of L-CGF by the Trainees on all other qualities using the Session

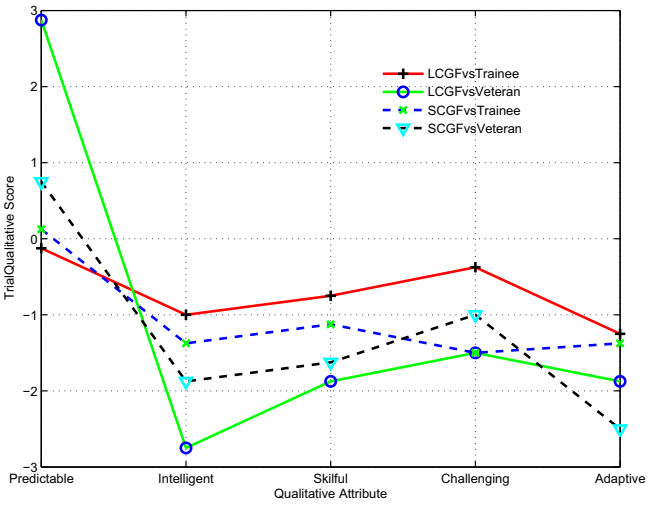

Fig. 21. Comparison of the qualitative assessment of the CGFs using the trial questionnaires.

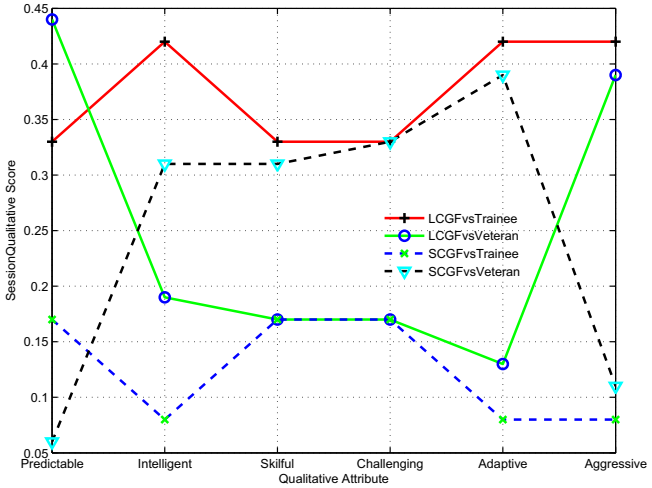

Fig. 22. Comparisons of the qualitative assessment of the CGFs using the session questionnaires.

questionnaires are consistent with their assessments of L-CGF using the Trial questionnaires. Also from Fig. 22, the assessments of L-CGF by the Veterans and the Trainees are consistent with the qualitative assessments illustrated in Fig. 21. However, the Veterans found their L-CGF to be more intelligent and as skillful and challenging to the S-CGF assessed by the Trainees.

Discussions: As presented in Section 8.3, the Trial and Session questionnaires are used to gather qualitative assessments on the performance of the CGFs from the human pilots. The objective of taking such an approach is to determine how consistent the human pilots are in their qualitative assessments of the CGFs. From the trial and session qualitative assessments of L-CGF, the Trainees are more inclined to match the desirable attributes to L-CGF than the Veterans. In contrast, the Veterans appear to have considered S-CGF to have more of the desirable attributes in both qualitative assessments. The differences in the qualitative assessments of L-CGF by the Trainees and Veterans may be attributed to a similar set of factors that account for the differences in the quantitative outcome of the CGF–Human experiments.

Some inconsistencies in the qualitative assessments of L-CGF and S-CGF by the Trainees and the Veterans are observed. Using the Trial questionnaires, the Trainees considered L-CGF to be less predictable than S-CGF but considered L-CGF to be more predictable than S-CGF using the Session questionnaires. The Veterans considered L-CGF to be more adaptive than S-CGF they have flown against using the Trial questionnaires but considered L-CGF to be less adaptive than the same S-CGF using the Session questionnaires. They also considered L-CGF to be less predictable than S-CGF in both qualitative assessments. This means the Veterans agreed with the Trainees on how predictable L-CGF was through their session qualitative assessments but not in their trial qualitative assessments. The Veterans also agreed with the Trainees on how adaptive L-CGF in their Trial questionnaires but not in their Session questionnaires. Consequentially, the Veterans and the Trainees had contrasting qualitative assessments of L-CGF on the remaining attributes with respect to S-CGF.

11. Conclusion

The detailed presentation of our studies shows the CGF–CGF experiments to be more trivial to the CGF–Human experiments in terms of the ability to show positive learning by the adaptive

CGF. Based on the quantitative results from the CGF–Human experiments, the adaptive CGF is more effective in out-maneuvering the trainee pilots than the veteran pilots. In both situations, the adaptive CGF turns out to be more challenging over time by scoring higher EqualMatch rates against the human pilots. Subsequent analysis of the qualitative assessments of the adaptive CGF by the Trainees and Veterans offers some rather contrasting viewpoints. Using blind questionnaire-based assessments, the Trainees were able to match the desirable attributes to the adaptive CGF whereas such qualities of the adaptive CGF were missed by the Veterans.

The differences in qualitative assessments by the Veterans and Trainees may be due, in part, to their ability to fly against the CGFs, their professional background and the use of different scenarios. The victories scored by the adaptive CGF may be attributed, in part, to some inconsistencies of the human pilots during 1-v-1 dogfights. This is evident from the performance of the Trainees and the Veterans against the doctrine-driven CGF. Despite of the inconsistencies, the human pilots are still able to eventually out-maneuver the adaptive CGF. This may mean the adaptive CGF is still not learning well and fast enough against the human pilots. Limitations on how the adaptive CGF can respond to the human pilots may also account for the negative performance outcome.

Further experiments on the CGF–CGF experiments shall include getting the adaptive CGF to respond to states using fundamentally different strategies such as being offensive and defensive at different times within the same session. Integrating FALCON into a broader cognitive architecture (Ng et al., 2010) may help in this aspect. Future work beyond 1-v-1 dogfight may involve formation flying and coordinated air combat (Virtanen, Hämäläinen, & Mattila, 2006). In addition, *future* simulator-based training may have adaptive CGF that can lead the trainees through different levels of training. Just like the human instructors, such adaptive CGF will have a sense of the skill level of the trainees and can moderate its own responses in real time.

Acknowledgments

This work is supported by the DSO National Laboratories under Research Grant DSOCL11258. This project was conducted in close collaboration with Khee-Yin How and his team at DSO National Laboratories, Seng-Beng Ho and his team at Temasek Laboratories @ NUS, Adrian Yeo and his team at CAE Singapore (S.E.A.) Pte. Ltd. and Sylvain Caron and his team at CAE (Montreal) Inc. This work is dedicated to the memories of Wallace Peel, an CAF veteran combat pilot from Sylvain's team who contributed invaluably to this work.

References

Ardema, M. D., & Rajan, N. (1987). An approach to three-dimensional aircraft pursuit-evasion. *Computers & Mathematics with Applications, 13*, 97–110.

Bell, H. H., & Waag, W. L. (1998). Evaluating the effectiveness of flight simulators for training combat skills: A review. *International Journal of Aviation Psychology, 8*, 223–242.

Bloom, P. C., & Chung, Q. B. (2001). Lesson learned from developing a mission-critical expert system with multiple experts through rapid prototyping. *Expert Systems with Applications, 20*, 217–227.

CAE Inc. (2007). CAE STRIVE™ CGF. Product Brochure.

Carpenter, G. A., & Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 54–115.

Cil, I., & Mala, M. (2010). A multi-agent architecture for modelling and simulation of small military unit combat in asymmetric warfare. *Expert Systems with Applications, 37*, 1331–1343.

Farmer, E., van Rooij, J., Riemersma, J., Jorna, P., & Moraal, J. (2003). *Handbook of simulator-based training*. Ashgate Publishing Limited.

Gilmore, J. F. (1985). Military applications of expert systems. *Future Generation Computer Systems, 1*, 403–410.

Gopher, D., Well, M., & Bareket, T. (1994). Transfer of skill from a computer game trainer to flight. *Human Factors: The Journal of the Human Factors and Ergonomics Society, 36*, 387–405.

Grefenstette, J. J., Ramsey, C. L., & Schultz, A. C. (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning, 4*, 355–381.

Grimm, W., Well, K. H. (1991). Lecture notes in control and information sciences. Chapter Modeling air combat as differential game: Recent approaches and future requirements. (pp. 1–13).

Heinze, C., Smith, B., & Cross, M. (1998). Thinking quickly: Agents for modeling air warfare. In *Advanced topics in artificial intelligence* (Vol. 1502, pp. 47–58).

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine, 20*, 27–42.

Louvieris, P., Gregoriades, A., & Garn, W. (2010). Assessing critical success factors for military decision support. *Expert Systems with Applications, 37*, 8229–8241.

McGinnis, M. L., & Phelan, R. G. (1996). A hybrid expert system for scheduling the U.S. army's close combat tactical trainer (CCTT). *Expert Systems with Applications, 11*, 157–176.

Nason, S., & Laird, J. (2005). Soar-RL: Integrating reinforcement learning with soar. *Cognitive Systems Research, 6*, 51–59.

Ng, G.-W., Tan, Y.-S., Teow, L.-N., Ng, K.-H., Tan, K.-H., & Chan, R.-Z. (2010). A cognitive architecture for knowledge exploitation. In *Proceedings of the 3rd conference on artificial general intelligence*.

Pausch, R., Crea, T., & Conway, M. (1992). A literature survey for virtual environments: Military flight simulator visual systems and simulator sickness. *Presence: Teleoperators and Virtual Environment, 1*, 334–363.

Pereira, R., Sanchez, J. L., & Rives, J. (1999). Knowledge-based maneuver and fire support planning. *Expert Systems with Applications, 17*, 77–87.

Rodin, E. Y., & Amin, S. M. (1992). Maneuver prediction in air combat via artificial neural networks. *Computers & Mathematics with Applications, 24*, 95–112.

Smith, C. A. P. (2004). Decision support for air warfare: Detection of deceptive threats. *Group Decision and Negotiation, 13*, 129–148.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Tan, A.-H. (2004). FALCON: A fusion architecture for learning, cognition, and navigation. In *Proceedings of the IJCNN* (pp. 3297–3302).

Tan, A.-H. (2007). Direct code access in self-organizing neural networks for reinforcement learning. In *Proceedings of the IJCAI* (pp. 1071–1076).

Tan, A.-H., Lu, N., & Dan, X. (2008). Integrating temporal difference methods and self-organizing neural networks for reinforcement learning with delayed evaluative feedback. *IEEE Transactions on Neural Networks, 19*, 230–244.

Teng, T.-H. (2012). Cognitive information systems for context-aware decision support. Ph.D. thesis, School of Computer Engineering, Nanyang Technological University.

Teng, T.-H., & Tan, A.-H. (2008). Cognitive agents integrating rules and reinforcement learning for context-aware decision support. In *Proceedings of the IAT* (pp. 318–321).

Teng, T.-H., & Tan, A.-H. (2012). Knowledge-based exploration for reinforcement learning in self-organizing neural networks. In *Proceedings of the IAT* (pp. 332–339).

Teng, T.-H., Tan, Z.-M., & Tan, A.-H. (2008). Self-organizing neural models integrating rules and reinforcement learning. In *Proceedings of the IJCNN* (pp. 3770–3777).

Teng, T.-H., Tan, A.-H., Ong, W.-S., & Lip, K.-L. (2012a). Adaptive CGF for pilots training in air combat simulation. In *Proceedings of the 15th international conference on information fusion* (pp. 2263–2270).

Teng, T.-H., Tan, A.-H., Tan, Y.-S., & Yeo, A. (2012b). Self-organizing neural networks for learning air combat maneuvers. In *Proceedings of the IJCNN* (pp. 2859–2866).

Teng, T.-H., Tan, A.-H., & Tan, Y.-S. (2012c). Self-regulating action exploration in reinforcement learning. *Procedia Computer Science, 13*, 62–74.

Virtanen, K., Hämäläinen, R. P., & Mattila, V. (2006). Team optimal signaling strategies in air combat. *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans, 36*, 643–660.

Wallis, P., Ronnquist, R., Jarvis, D., & Lucas, A. (2002). The automated wingman – using JACK intelligent agents for unmanned autonomous vehicles. In *Proceedings of the IEEE aerospace conference* (Vol. 5, pp. 2615–2622).

Wang, T.-C., & Chang, T.-H. (2007). Application of topsis in evaluating initial training aircraft under a fuzzy environment. *Expert Systems with Applications, 33*, 870–880.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*, 279–292.

Wray, R. E., Laird, J. E., Nuxoll, A., Stokes, D., & Kerfoot, A. (2005). Synthetic adversaries for urban combat training. *AI Magazine, 26*, 82–92.

Yin, Y., Gong, G.-H., & Han, L. (2011). Experimental study on fighters behaviors mining. *Expert Systems with Applications, 38*, 5737–5747.