Un modello 3D di robotica evolutiva per lo sviluppo di controller autonomi per robot volanti

Fabio Ruini (corresponding author)
Angelo Cangelosi
Adaptive Behaviour and Cognition Research Group, Centre for Robotics and Neural Systems, University of Plymouth, Drake Circus, PL4 8AA, Plymouth, United Kingdom
Tel. +44 (0)1752 586288, Fax +44 (0)1752 586200
fabio.ruini@plymouth.ac.uk

1. Introduzione

Il presente lavoro descrive alcuni risultati preliminari ottenuti dagli autori nell'estensione di un modello simulativo da loro precedentemente sviluppato (Ruini, Cangelosi & Zetule, 2009) (Ruini & Cangelosi, 2009). Il modello in questione e' mirato all'evoluzione di controller autonomi, basati su reti neurali, per squadre di MAVs (Micro-unmanned Aerial Vehicles). L'approccio utilizzato per lo sviluppo di tali controller si basa su una combinazione di robotica evolutiva (Nolfi & Floreano, 2000) e sistemi multi-agente (Wooldridge, 2009).

2. Panoramica del modello 2D

Il modello simulativo originario del quale trattiamo qui un'estensione si sviluppa in un ambiente bidimensionale. Al suo interno e' prevista la presenza alternata di "squadre" composte da 4 velivoli ciascuna, impegnate nello svolgimento di un'operazione comune. All'inizio di ogni epoca di test, un target viene dislocato in una certa posizione all'interno di questo ambiente. Nei vari scenari, basandosi su un mix di informazioni di carattere "globale"

ed altre raccolte localmente, i MAVs, guidati da un controller neurale "embedded", devono riuscire a navigare fino al target e, una volta giunti in sua prossimita', mettere in atto una determinata operazione (rappresentata dall'attivazione di un particolare neurone booleano di output).

Il modello in questione presenta alcune caratteristiche innovative per quanto riguarda le simulazioni di robotica evolutiva. In particolare e' presente una componente legata al moto degli agenti che raramente si trova negli esperimenti in letteratura. Nello scenario delineato qui sopra i MAVs devono infatti mantenere una velocita' costante per tutta la durata della loro vita e non possono mai entrare in contatto con alcun ostacolo (sia esso un edificio, un altro velivolo o uno dei confini dell'ambiente), pena l'immediato fallimento del test nel quale sono impegnati. Questa condizione li costringe ad una pianificazione estremamente accurata (e "conservativa") di ciascun movimento: compito tantopiu' problematico se si considera il fatto che essi non sono in possesso di alcuna mappa, ne' predefinita ne' costruita in maniera dinamica, dell'ambiente di riferimento. Anche il modo in cui i MAVs devono operare una volta raggiunto il target rende il compito che essi devono svolgere particolarmente impegnativo. Essi non possono infatti attivare il succitato neurone booleano piu' di una volta, dato che il suo utilizzo provoca l'immediata "morte" dell'agente.

Durante il processo evolutivo, nonostante le costrizioni di cui sopra, accanto alle capacita' standard di navigazione sono emersi in maniera relativamente semplice comportamenti di piu' alto livello. Sono stati studiati nel dettaglio quattro scenari: (1) ambiente sgombro da ostacoli; (2) ambiente con ostacoli (disposti in maniera tale da ricreare un'area urbana ispirata al Canary Wharf di Londra); (3) target in movimento, in grado di avvertire l'avvicinarsi di un MAV e tentare quindi di allontarsi da esso; (4) target fisso (a) o in movimento (b) ed azione finale che richiede cooperazione tra i membri del team (due MAVs devono approcciare il target ed attivare il loro neurone booleano di output in rapida successione). In tutti questi casi, il processo evolutivo ha condotto a controller in grado di svolgere il compito richiesto, seppur con perfomance diverse a seconda del livello di difficolta' del compito. Nel dettaglio, la percentuale complessiva di test completati con successo per gli individui dell'ultima generazione e' stata rispettivamente del 93.46% nello scenario 1, dell'87.18% nello scenario 2, del 76.4% (calcolato come media tra 5 simulazioni dove il target si muoveva ad altrettante differenti velocita') nello scenario 3, del 72.3% nello scenario 4a ed infine del 49.6% nello scenario 4b.

3. Il passaggio al 3D

Il passaggio ad un modello 3D presuppone l'aggiunta di due gradi di liberta' (DOFs) agli agenti rispetto a quelli disponibili nel simulatore 2D (dove i MAVs potevano soltanto ruotare il loro corpo in senso orario o anti-orario, basandosi di fatto su un singolo DOF). Se consideriamo un sistema di riferimento ortogonale incentrato nel centro di massa dei velivoli simulati e che si muove con essi, le rotazioni che i MAVs possono effettuare sono tre: yaw, pitch e roll. Nel dettaglio, lo yaw e' una rotazione dell'aereo lungo la sua asse verticale, mentre pitch e roll si riferiscono rispettivamente alle rotazioni che avvengono lungo l'asse che collega le due ali, e quelle attorno all'asse che va dalla coda al muso (vedi Figura 1).

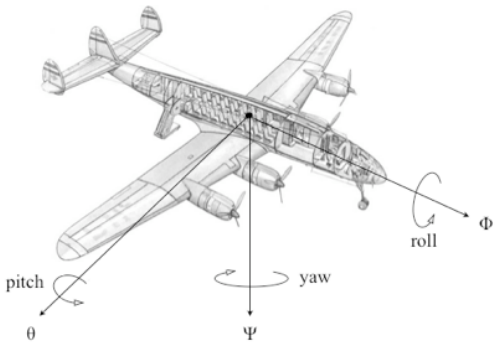

Figura 1. I gradi di liberta' solitamente associati ad un velivolo ad ali fisse

Il controller di ciascun MAV e' implementato mediante una rete neurale feed-forward. Le informazioni che essa riceve in input sono: (1) la distanza dal target, (2) l'angolo orizzontale tra target e MAV relativo alla direzione in cui quest'ultimo sta puntando, (3) la differenza di altitudine tra target e velivolo, l'angolazione corrente di roll (4) e pitch (5) dell'aereo. In output sono presenti neuroni continui che modificano in maniera incrementale la rotazione del MAV lungo le tre assi, oltre al neurone booleano che implementa la "azione finale" che i velivoli devono mettere in atto una volta raggiunto il target.

L'evoluzione verso reti neurali in grado di esibire il comportamento desiderato e' garantita dall'utilizzo di un algoritmo genetico (Mitchell, 2000).

Una popolazione iniziale di 100 controller e' creata con pesi sinaptici e bias assegnati casualmente nel range [-10.0; +10.0]. Ciascun controller e' assegnato in maniera univoca ad un intero team di MAVs: ogni velivolo appartenente a tale squadra sara' pertanto guidato dalla medesima rete neurale. I team sono formati da 4 MAVs e vengono testati per 12 epoche (con il target dislocato in altrettante posizioni diverse) con i velivoli inizialmente distribuiti in prossimita' dei quattro angoli dell'ambiente e dotati di una certa riserva di energia (la quale decresce con il passare del tempo, provocando la caduta del MAV in caso di esaurimento). L'ambiente di riferimento e' sgombro da ostacoli ed il setup corrisponde pertanto allo scenario 1 del simulatore 2D. Una volta che tutti i team sono stati messi alla prova e valutati attraverso una funzione di fitness[1], i migliori 10 vengono selezionati per la riproduzione. Ciascuno di essi genera 9 copie del suo genoma, sui cui pesi e bias e' applicata con probabilita' 0.15 una mutazione casuale che ne modifica il valore di un ammontare casuale nel range [-0.2; +0.2]. L'operatore di elitismo e' applicato al fine di preservare il miglior controller di ciascuna generazione. 9 nuovi controller, con pesi e bias assegnati in maniera casuale, vengono creati ad ogni generazione. L'intero processo viene iterato per 10,000 generazioni e ripetuto per 20 volte, al fine di limitare il piu' possibile gli effetti dovuti al caso.

Varie architetture di rete sono state testate (si veda il riassunto in Tabella I). Si tratta di diverse topologie di reti feed-forward, che possono essere dotate o meno di uno strato intermedio e che ricevono in input informazioni in valore continuo o discretizzate. Le architetture differiscono anche per come il roll e' stato implementato (esso puo' essere assente, generato come effetto secondario dello yaw oppure utilizzato in maniera indipendente), nonche' per la presenza di memorie di breve termine, implementate sotto forma di reti di Elman (Elman, 1990) o di Jordan (Jordan, 1986) a seconda dei casi.

Tabella I. Le architetture di rete utilizzate nel simulatore 3D

Arch. NN	Input	Strato nascosto	Roll	Memoria
1	D	No	No	No
2	C	No	No	No
3	D	No	Con yaw	No

[1] La funzione di fitness e' data dalla semplice formula: $f = -\alpha + \beta$, dove α e' la distanza, calcolata come media per i 12 test, tra il target ed il velivolo del team che piu' vi e' arrivato vicino nel momento in cui ha attivato il suo neurone booleano di output; β e' invece l'ammontare medio di energia rimasta nei serbatoi dei MAVs che hanno portato a termine con successo la missione.

4	C	No	Con yaw	No
5	D	No	Indipendente	No
6	C	No	Indipendente	No
7	D	Si'	Con yaw	No
8	C	Si'	Con yaw	No
9	D	Si'	Indipendente	No
10	C	Si'	Indipendente	No
11	D	No	Con yaw	Jordan
12	C	No	Con yaw	Jordan
13	D	Si'	Con yaw	Elman
14	C	Si'	Con yaw	Elman
15	D	No	Indipendente	Jordan
16	C	No	Indipendente	Jordan
17	D	Si'	Indipendente	Elman
18	C	Si'	Indipendente	Elman

4. Risultati preliminari in 3D e confronto con il 2D

I risultati ottenuti con il nuovo simulatore sono riassunti in Tabella II.

Tabella II. Risultati ottenuti con il simulatore 3D per le varie architetture di rete

Arch. NN	Fitness media	Fitness massima	Percentuale successi
1	126.5897	461.002	75.49
2	168.2282	454.7155	76.65
3	-130.3004	345.6638	43.63
4	-35.2221	407.9116	54.93
5	53.7617	406.5243	60.26
6	5.4618	391.213	57.49
7	-82.8986	360.323	47.54
8	49.8237	438.7521	63.96
9	119.3603	444.415	67.55
10	53.28	420.1875	60.28
11	-221.9481	232.8541	20.86
12	-160.9756	256.3188	24.72
13	-258.8055	113.8254	12.95
14	-265.2192	130.6172	15.5

15	-42.8824	311.3557	35.81
16	-18.4841	346.6911	45.3
17	-92.2134	291.5339	33.46
18	-85.5089	296.8809	34.21

Cosi' come era atteso, le architetture 1 e 2 sono quelle che hanno in assoluto generato i risultati migliori. Si tratta pero' di due simulazioni di benchmark, che, non implementando in nessun modo il roll, generano movimenti dei MAVs non realistici[2]. Aspetto interessante e' che i controller sembrano funzionare meglio quando possono gestire in maniera indipendente il roll, rispetto a quando questo e' legato allo yaw. Il tutto nonostante la gestione diretta di questa componente aumenti il livello di complessita' della rete e le dimensioni dello spazio delle soluzioni. Le reti multi-strato dotate di uno strato intermedio hanno fatto registrare performance generalmente migliori rispetto a quelle dove lo strato di input era direttamente connesso a quello di output. L'introduzione di ricorrenze, a parte aver rallentato in maniera significativa il processo evolutivo, non ha condotto ad un miglioramento delle prestazioni. Piu' ambiguo e' invece il ruolo giocato dalla discretizzazione dell'input sensoriale, che in alcuni casi ha determinato vantaggi rispetto all'utilizzo di input continui, mentre ha condotto in altre circostanze a deterioramenti delle performance.

5. Multi-threading

Il maggior livello di complessita' derivante dall'utilizzo di un simulatore 3D ha stimolato l'investigazione sugli effetti prodotti dal ricorso a procedure di programmazione multi-threading. Il codice del motore evolutivo (che e' stato scorporato dalla parte grafica in maniera tale da alleggerire il carico di lavoro richiesto durante l'evoluzione) e' stato pertanto modificato in maniera tale da poter sfruttare tutta la potenza di calcolo disponibile sui calcolatori in uso[3], distribuendo i team di MAVs da testare sui vari core a disposizione.

I benefici che questa modifica ha apportato, per quanto positivi, sono stati inferiori rispetto alle nostre attese. Quello che ci aspettavamo era che il passare dall'utilizzo di un singolo core a 8 in contemporanea avrebbe com-

[2] Il simulatore 3D, allo stato attuale, non si basa su alcun engine fisico, ma l'idea alla base e' quella di far si' che i MAVs esibiscano comunque movimenti che non siano in palese contrasto con le principali leggi della fisica.

[3] Le simulazioni sono state eseguite su quattro Apple Xserve, ottenuti da Apple nel contesto del programma ARTS (http://www.apple.com/uk/education/hed/arts/). Ciascuna di queste macchine e' mossa da un doppio processore Intel Xeon Quad-Core a 2.8GHz. Il multi-threading e' stato ottenuto sfruttando le apposite funzioni offerte dalle librerie Qt (http://qt.nokia.com).

portato un corrispondente incremento di almeno il 600% della velocita' di esecuzione del codice (o comunque un valore leggermente inferiore rispetto al teorico +700%, per via di qualche inevitabile overhead). In realta', le analisi effettuate hanno mostrato come il miglioramento di performance abbia luogo soltanto con una magnitudo piu' ridotta. Sono stati condotti 10 esperimenti, nei quali sono state evolute 50 generazioni per l'architettura neurale 9, sia in single che in multi-threading. Nel primo caso, portare a termine il test ha richiesto una media di 150,354.8 msec; nel secondo caso, invece, 27,765.1 msec. L'incremento di velocita' nel passaggio al multi-threading e' sicuramente significativo (+441.52%), ma ben lontano rispetto al teorico +700% ottenibile.

6. Conclusioni

In questo articolo e' stata descritta la prima parte di un lavoro mirato all'estensione del pre-esistente simulatore 2D in un nuovo modello 3D, nonche' descritta quella che e' stata l'esperienza degli autori nell'utilizzo di metodologie di programmazione multi-threading. Il livello di realismo del precedente modello e' stato incrementato e questi sviluppi costituiscono pertanto un passo aggiuntivo lungo la strada verso il possibile trasferimento dei controller sviluppati in simulazione su piattaforme robotiche reali.

Malgrado un generale decremento delle performance dei MAVs nello svolgere compiti che nell'analogo simulatore 2D venivano portati a termine in maniera relativamente semplice, il processo evolutivo ha comunque condotto a risultati positivi anche per quanto riguarda il nuovo modello. Il peggioramento delle prestazioni era un risultato atteso in quanto l'aggiunta di nuovi gradi di liberta' al simulatore aumenta notevolmente le dimensioni dello spazio delle soluzioni che l'algoritmo evolutivo deve esplorare alla ricerca di un'appropriata configurazione di pesi sinaptici e bias per i controller.

I risultati presentati in relazione al multi-threading non sono sufficientemente solidi e tantomeno generalizzabili al punto da permettere di trarre da essi conclusioni definitive. Questi suggeriscono pero' che il ricorso a metodologie di programmazione parallela debba essere pianificato con attenzione e criterio, in quanto non necessariamente conduce verso i miglioramenti attesi nella velocita' di esecuzione delle simulazioni. Nel caso specifico, data la relativa velocita' nell'effettuare la valutazione di un singolo team, una possibile soluzione potrebbe essere quella di modificare il codice in maniera tale da distribuire su ciascun core disponibile la valutazione di un maggior numero di squadre, riducendo in questo modo l'inevitabile impatto dell'overhead. In aggiunta alla replica degli esperimenti effettuati con il simulatore 2D, parte dei lavori futuri sara' dedicata ad approfondire questo aspetto.

Disclaimer

Efforts sponsored by the Air Force Office of Scientific Research, Air Office Material Command, USAF, under grant number FA8655-07-1-3075.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies and endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon.

Bibliografia

1. Ruini, F., Cangelosi, A., Zetule, F.: Individual and Cooperative Tasks performed by Autonomous MAV Teams driven by Embodied Neural Network Controllers. In: Proceedings of the 2009 International Joint Conference on Neural Networks, pp. 2717-2724 (2009)
2. Ruini, F., Cangelosi, A.: Extending the Evolutionary Robotics Approach to Flying Machines: an Application to MAV Teams. Neural Networks, 22, 812—821 (2009)
3. Nolfi, S., Floreano, D.: Evolutionary Robotics. MIT Press, Cambridge, MA (2000)
4. Wooldridge, M.: An Introduction to Multiagent Systems. John Wiley & Sons, Hoboken, NJ (2009)
5. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA (1998)
6. Jordan, M.I.: Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. In: Proceedings of the Eighth Annual Conference of the Cognitive Science Society, pp. 531–-546 (1986)
7. Elman, J.L.: Finding Structure in Time. Cognitive Science, 14, 179--211 (1990).