Evolving ANNs for Spacecraft Rendezvous and Docking

J. Leitner*, C. Ampatzis*, D. Izzo*

*Advanced Concepts Team, European Space Agency
e-mail: {jurgen.leitner, christos.ampatzis, dario.izzo} @esa.int

Abstract

We here approach the problem of designing a controller for *automatic rendezvous and docking* (AR&D). As a first step towards a fully reactive neurocontroller (based on state feedback), an artificial neural network is trained offline by a fitness based genetic algorithm to fulfill the docking task. Its performance using one specific docking case, with fixed initial and boundary conditions, is compared to a numerical solution of the corresponding optimal control problem. The concept of a "stop-neuron" is introduced here to the neurocontroller for the first time to co-evolve the time taken to achieve the desired docking position.

1 Introduction

Intelligent machines, or agents, are systems that try to maximize their chance of success through adaptation and learning to take actions based on their perception. The key to designing successful artificial intelligent entities, not just for space, are adaptability and autonomy. The operation of intelligent machines in space has not yet been extensively tested, it could though lead to research on novel, far more flexible and adaptive approaches to computing and control, with the efficiency and precision needed for space. There is desire to increase the operational capabilities and avoid (or self-adapt to) failures on spacecraft, satellites, rovers and other agents in space, by being able to autonomously make decisions.

Space systems, unlike most robotics systems, usually require extreme precision (in the task execution) and are much more affected by simulation/reality discrepancies. On the other hand full state information (e.g. position via the Global Positioning System) can often be assumed and sensory noise is typically smaller than the one usually considered in (evolutionary) robotics research. Neurocontrollers can be seen as black-boxes translating inputs (including position, attitude and velocities) to control outputs (thruster activations), thus providing a state-feedback that can be used in a closed loop control. They are an alternative to hand-coded control laws (difficult to design) or optimal control strategies (lacking in robustness and adaptiveness).

Using the well understood theory of optimal control analytical solutions to optimal control problems can only be obtained for some simple problems (see Pontryagin [13]), e.g. linear systems. The theory delivers the solution to the control synthesis problem, expressing the optimal control directly as a function of the state (*state-feedback*) thus indirectly also solving the Hamilton-Jacobi-Bellman equations [5], which are a *sufficient and necessary* condition for optimality. In most cases though, the Pontryagin Maximum Principles allows to find *necessary* conditions for optimality. When exploited in a numerical algorithm to find the optimal control (e.g. a single shooting method) it then returns the optimal control history as a function of time.

In short, numerical techniques derived from the theory of optimal control allow to find $\mathbf{u}^*(t)$, an open loop strategy that is bound to fail in an uncertain and disturbed environment. The vision, and basic motivation for this research, is to find a *state-feedback*, that is $\mathbf{u}^*(\mathbf{x})$, providing more adaptiveness and robustness to the controller. We propose neurocontrollers as a possible solution to approximate $\mathbf{u}^*(\mathbf{x})$. It is also of great interest to compare the 'price' one has to pay in terms of optimality when using this (or other reactive control) techniques.

This paper presents preliminary results of our ongoing effort to evolve a neurocontroller for an automatic rendezvous and docking task. This problem is split into the following:

(a) We aim at reproducing the optimal control with an neurocontroller, i.e. using $\mathbf{u}^*(\mathbf{x})$ instead of $\mathbf{u}^*(t)$, for a selected docking case, given a fixed initial position and boundary conditions

(b) The neurocontroller is then trained from multiple initial conditions, leading, most likely, to a hiogher degree of sub-optimality but allowing for generalization, i.e. a 'price' is paid in exchange for robustness and adaptability

This paper is presenting preliminary results on **(a)**, designing a neurocontroller to reproduce the optimal control.

2 Related Work

The work here builds on previous research done in the area of neurocontrollers and evolutionary robotics [12] as well as the research and progress done in automatic rendezvous and docking (AR&D).

ANNs, when used as a controller, are usually referred to as *neurocontrollers*. The inputs of the network are describing the state or sensory inputs (of the agent) and the outputs are the behaviour or motor control. These neurocontrollers have been increasingly used in robotics (see above description of evolutionary robotics), e.g. recently in cooperative robotics [2], and also in space applications.

Space robotics, due to the possibly highly unknown environment, as well as the high cost involved in preparation, launch and operation on space missions, is a field where robust and adaptive control are highly relevant. Neurocontrollers have been shown to be useful in space, from the development of stable controllers [15], adaptive controllers [10] to basic feasibility tests for docking cases [7]. Results applying evolved neurocontrollers for low-thrust trajectories [4] has recently shown with renewed strength the plausibility of using these controllers in space.

3 Problem Formulation

In this section we describe the details of the docking task considered, the dynamic environment and the spacecraft actuation capabilities.

3.1 Environment Equations

The dynamics of the problem are represented by the Hill-Clohessey-Wiltshire (HCW) equations, describing the relative motion of spacecraft in orbit (for a short introduction see [8]), with the addition of spacecraft attitude. This leads to the following two-dimensional dynamic system:

$$\ddot{x} = 2n\dot{y} + 3n^2 x + (u_l + u_r)\cos\theta \tag{1}$$

$$\ddot{y} = -2n\dot{x} + (u_l + u_r)\sin\theta \tag{2}$$

$$\ddot{\theta} = (u_l - u_r)\frac{1}{mR} \tag{3}$$

where θ, m and r are the attitude, mass and radius of the spacecraft respectively. A constant $mR = 0.75$ is used throughout the paper. To describe the relative motion in a circular orbit around earth in a typical LEO altitude $n = 0.08$ is assumed (stemming from the distance R and the gravitational parameter μ).

This environment is different from the one usually used in evolutionary robotics, e.g. the forces acting on a thrust-free agent are (most of the time) not zero and are different depending on the position in the environment. This is shown in Figure 1, where these forces are shown as simulated trajectories of non-controlled spacecraft at various starting positions (9x9 grid) over 2 seconds. Therefore the simple task of phototaxis can, in orbit, not be solved efficiently (accounting for fuel efficiency) by just pointing and moving towards the light in a straight line, and a more complex phototactic behaviour is needed [8].

For a more detailed description of the docking problem, the current approaches and its challenges the reader is referred to [6].

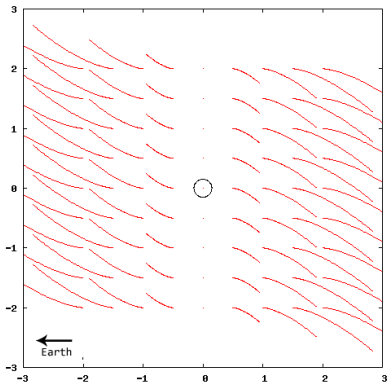

Figure 1. The gravitational environment, using $n = 0.08$. The equilibrium points are seen on the Y-axis.

3.2 Spacecraft Model

The spacecraft is modeled as a circle with radius r and mass m and has two thrusters on opposite sides aligned to the same body line. These thrusters can be fired independently, in both directions and within the range of maximum allowed thrust ($0.1N$). Negative thrust in the model relates to thrusting in the opposite direction (i.e. employing a breaking maneuvre). The thrust levels are the control output u_l and u_r for the left and right thrusters respectively.

3.3 Spacecraft Rendezvous and Docking

The AR&D process involves a series of maneuvers and controlled trajectories, which successively bring the active vehicle (usually referred to as *chaser*) into the vicinity of, and eventually into contact with, the passive spacecraft (*target*). The guidance, navigation and control (GNC) system of the chaser controls the state parameters required for entry into the docking interfaces of the target vehicle and for capture.

4 Neurocontroller

An artificial neural network with a feed-forward architecture (see Figure 2) was chosen to control the spacecraft because of their parallel, analogue, fault-tolerant and

adaptive nature [14]. In particular, a multi-layer perceptron is directly controlling the two thrusters u_l and u_r. The network consists of 6 input neurons receiving input from the full state information, including position, speed, attitude and angular velocity $(x, \dot{x}, y, \dot{y}, \theta, \dot{\theta})$. The hidden layer contains from 10 to 30 neurons, depending on the task, and the output layer controls the thrusters.

The particularity of our implementation is the inclusion of a "stop-neuron", an extra output neuron that, when above a certain threshold, would indicate that the simulation is to be stopped. This means, that even if a trial has a certain maximum length (in time), the neural network is able to terminate it at any point in time before the limit. That is, the network is free to decide according to its inputs if the trial should be terminated, leading to the evaluation of the performance of the spacecraft. Ideally, the network should "learn" to activate this neuron when the docking conditions have been met. The inclusion of a stop-neuron allows the neural network to directly control not only the thrusters of the spacecraft but also the time duration of the event. To the best of our knowledge, this is the first attempt to optimize position and time in a robotic task, by utilizing an ANN directly controlling actuators and the time duration. The stop-neuron (highlighted in Figure 2) is, like the other output neurons, fully connected to the hidden layer.

4.1 Fitness Function

The fitness function driving the artificial evolution is rewarding smaller distance to the goal, smaller speed of the spacecraft and an attitude matching the desired value at the final position.

The fitness function allows to treat the evolution as a maximization problem, including the attitude ϕ, distance d and speed s of the spacecraft, it also includes a time bonus, based on the time needed for the whole docking maneuvre (t_d), if the constraints are satisified:

$$f = \begin{cases} 1 + \frac{t_{max} - t_d}{t_{max}} & \text{if within constraints} \\ \\ \frac{1}{(1+\phi)(1+d)(1+s)} & \text{otherwise} \end{cases} \tag{4}$$

where t_{max} is the maximum time the simulation waits for the stop-neuron to fire. If at that time the stop neuron is not fired the simulation stops and evaluates the neurocontroller at t_{max}. This setup allows to optimize the controller by minimizing the parameters ϕ, d and s. A reasonable guess for the maximum time is needed, since it influences the possibilities of the control but as a draw-back also the running time of the simulation. A good guess we found is to take about twice the value found for t when solving the optimal control problem.

This two-step approach allowed for faster evolution towards a good neurocontroller by smoothing the jump between first evolving a docking behaviour and then minimizing time.

The mentioned constraints are given by the boundary conditions of the docking problem, docking is assumed to happen if the following is met (ϕ_s and ϕ_d denote the attitude of the spacecraft and the attitude desired at the docking position respectively):

$$|\phi_s - \phi_d| < \pi/8, \quad |d| < 0.1, \quad |s| < 0.1 \tag{5}$$

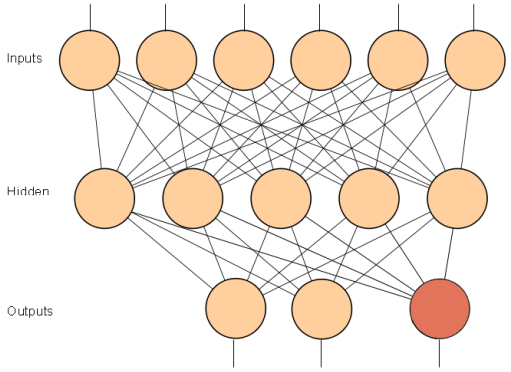

Figure 2. The schematics of one (simple) ANN as used for the neurocontroller.

5 Autonomous Docking Implementation

Autonomous docking is defined as one spacecraft approaching another one autonomously, from close proximity, with hard constraints on the final position, the final speed and the final attitude of the spacecraft (see Eq. 5). We are using an ANN representation and we evolve its weights with artificial evolution using PaGMO, an open source software platform for massively parallel global optimization developed at the Advanced Concepts Team (ACT) at the European Space Agency (ESA) [3], and its built-in simple genetic algorithm (SGA).

5.1 Evolving the Controller

A simple genetic algorithm (SGA) is used as a meta-heuristic optimizer to evolve new neurocontrollers. It tries to optimize a vector, representing the weights in the ANN, by defining a fitness value for each. It then tries to (intelligently) modify the vector to generate better (higher) fitness values. This is only one of the many approaches to evolve neurocontrollers for robots, a very good review of options is available in [12].

The SGA provides a basic implementation of a genetic algorithm using a floating point (not binary) encoding and some common mutation and crossover strategies.

For the current problem a random mutation was selected, with an exponential crossover and a roulette selection. The algorithm available in PaGMO works on single objective, box constrained problems. The mutation operator acts differently on continuous and discrete variables, though in this case we only use the continuous implementation.

5.1.1 Parameterization

The implementation of the docking problem is aiming for high configurability by parameterization. The parameter values chosen to evolve a neurocontroller for the docking problem, from one fixed starting condition, given the docking constraints defined above, are shown shown in Table 1.

For the docking problem we use a population of 50 individuals, each consisting of a vector of 103 elements. The elements of the vector represent the weights and nodes (6 input, 10 hidden and 3 output nodes) in the neural network and are in the range $[-10.0, +10.0]$. The SGA itself can be parametrized by the following:

- *number of generations* the SGA goes through at every iteration;
- *crossover probability* is the probability with which an element in the vector is taken from the father or the mother vector, i.e. the crossover starts at a random position and continues until a random variable is over the (probability) threshold;
- *mutation probability* describes the threshold value for generating a random value in the vector, i.e. a random value is overwriting the vector value if the random variable is over the (probability) threshold.;
- *elitism* defines the number of generations after which the best individual is reinserted into the population.

Table 1. Parameters chosen for the docking problem optimization run.

Population:	50
Input Neurons:	6
Hidden Neurons:	10
Output Neurons:	3
Number of Generations:	11
Crossover Probability:	0.95
Mutation Probability:	0.01
Elitism:	1
Stop-Neuron Threshold:	0.99
Maximum Time:	15
Integration Step:	0.1

Apart from these parameters, defining the simple genetic algorithm, the current implementation can further be detailed using the following parameters:

- *docking constraints:* varying the speed, distance and attitude constraints
- *positions:* the number of starting positions each neurocontroller is evaluated with
- *positioning strategy*: defines the randomness in the starting positions, for runs with multiple starting positions
- *fitness function:* allows to choose which fitness function is used to evaluate the neurocontroller
- *evolution stuck:* defines a parameter, after which if no improvement is detected, the population is reinitialized

The *"stop-neuron"* described earlier basically tells at which point of the simulation the individual is evaluated with the fitness function, i.e. its fitness is determined by the fitness function given above. The threshold of firing can be set by setting the timeneuron-threshold parameter.

The *"integration-step"* is described in the simulation section. It defines how detailed the simulation run is, with the drawback of longer runtimes for more detailed simulation.

5.1.2 The Simulation

A Runge-Kutta (RK4) numerical integrator is used to propagate the HCW equations (Eq. 1-3) using the neurocontroller outputs as values for u_l and u_r and feeds the state information after every integration-step interval (as parametrized) back into the neurocontroller. The value of the third output neuron is checked for whether it exceeds the defined threshold, and if so the simulation run stops and the fitness of the current individual for the specific run is determined.

5.2 Optimal Control

For the purpose of comparing the neurocontroller we here consider the system described by Eq. 1-3 and the problem of optimally steering the spacecraft, in a minimum time, from x_0, y_0, θ_0 to a final state satisfying the terminal conditions given in Eq. 5, as an optimal control problem. In this case the system hamiltonian is,

$$\begin{aligned}\mathcal{H} &= \lambda_x v_x + \lambda_y v_y + \lambda_\theta \omega \\ &+ \lambda_{v_x}\left[2nv_y + 3n^2 x + (u_l + u_r)\cos\theta\right] \\ &+ \lambda_{v_y}\left[-2nv_x + (u_l + u_r)\sin\theta\right] \\ &+ \lambda_\omega (u_l - u_r)\end{aligned}$$

which has some nonlinear terms in the state (i.e. θ) but is linear in the control.

We thus know that the optimal solution for this type of system will be $u_l^*, u_r^* = \pm u_{max}$ during non-singular arcs. The control will not necessarily be of this bang-bang type, due to the non-linearities of the rotational dynamics, giving possibility to yield singular arcs during a particular optimal trajectory. To obtain a solution to the optimal control problem many numerical algorithms exist. We here use a direct method [16] based on the impulsive transcription of the control, implemented using a modelling language for mathematical programming *AMPL*.

6 Results

Here the first, preliminary results are presented, aiming to compare the neurocontroller, trained for one single, fixed starting position, with the optimal control solution.

6.1 The Evolution Run

To evolve the neurocontroller a population of controllers is created and all individuals in the population are tested to determine their respective fitness. The SGA acts on the population, replacing individuals with new ones to optimize the best individual. In Figure 3 the fitness of the best individual in the population at each generation of one evolution run is plotted. Jumps of the best fitness are only seen during the beginning of the evolution process and the increase slows down during higher iterations, as seen in this representative plot.

The low mutation rate and the elitism of 1 ensure that the best solution stays within the population and allows for continuous improvement of the fitness. The payoff is an algorithm with no big variation and jumps and therefore a longer runtime.

6.2 Generated Trajectory

The trajectory, generated using the parameters in Table 1 and using the outputs of the neurocontroller in the numerical integrator, is logged and plotted and can be seen in Figure 4. The trajectories obtained using optimal control techniques (shown in the upper row) are compared with the one generated by the best neurocontroller[1] in the lower row. The figure includes (from left to right): the trajectory (XY-plot), the thruster activation (u_l in green and u_r in red) and the orientation of the spacecraft during docking.

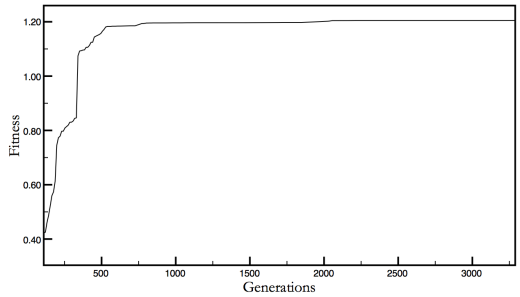

Figure 3. Typical improvement of the best fitness during an evolution run.

According to the fitness function definition given above, the best neurocontroller has a fitness of $f = 1.34$ for this specific docking problem and takes $9.9s$ to reach the target. In comparison, the optimal control yields a fitness of $f = 1.47$ due to the faster docking time of $7.9s$. A comparison of the key values obtained in these results is shown in Table 2. The numbers there show that the neurocontroller needs a higher total thrust, mainly because of the non-optimal firing and therefore longer time until docking (a plus of around 17% in total thrust vs. a plus of 25% in time for docking).

The trajectories itself, the first plots in the figure, look much alike, though the neurocontroller reaches a larger deviation in the Y axis during the approach but seems to come in closer (in X axis) to the final position at the end. The difference in values is most likely a result of the stop-neuron, which is used as a stop condition in the simulation for the case of the neurocontroller, whereas the optimal control stops as soon as the requirements (Eq. 5) are met.

The thruster activation, the plot in the middle column of Figure 4, of the optimal control shows the use of a singular arc, as described above, and a bang-bang like activation around it. The neurocontroller in comparison does not fully yield a bang-bang based solution and the singular arc is longer and almost extends to twice the time (and 5 times the amplitude).

The third column in Figure 4 shows the orientation of the spacecraft during the docking process. The optimal control solution and the neurocontroller look quite similar, suggesting that the neurocontroller follows a similar strategy as the optimal control. The curve is stretched be-

[1] `http://atlas.estec.esa.int/ariadnet/node/318`

Table 2. Comparing the results from the optimal control with the ones obtained by the neurocontroller.

	Optimal Control	Neurocontroller
Fitness	1.47	1.34
Total Thrust	1.21 N	1.41 N
t_d	7.9 s	9.9 s
θ_f	0.196	0.000
d_f	0.100	0.010
s_f	0.100	0.100

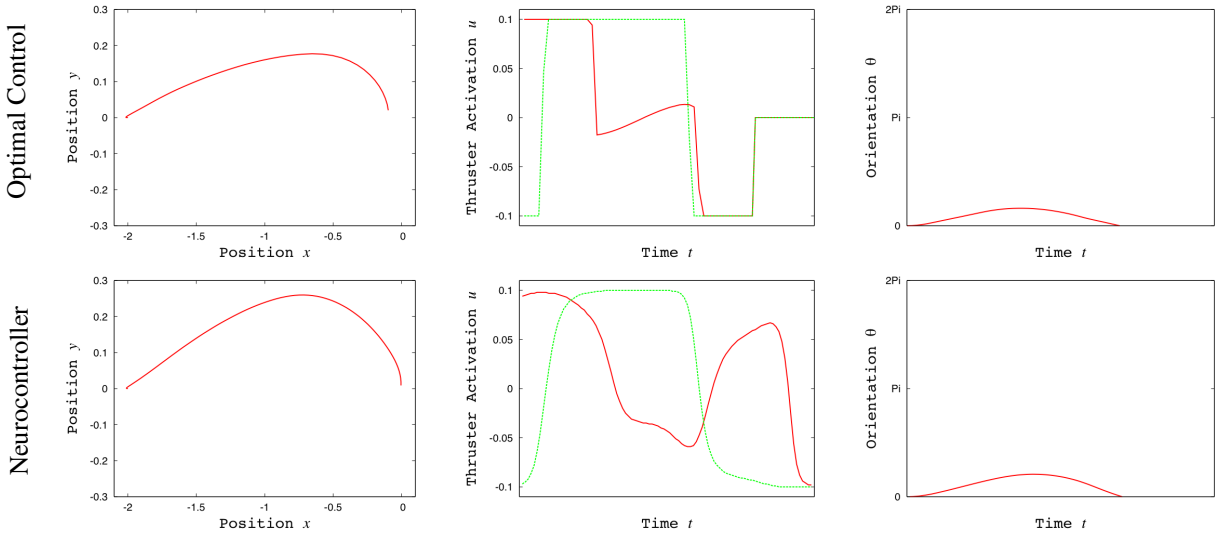

Figure 4. The trajectory of a spacecraft docking (from $(-2, 0)$) with another craft at origin generated by the optimal control and a neurocontroller. Thruster activations for both, u_l (green) and u_r (red), are shown.

cause of the somewhat less efficient use of the thrusters by the evolved controller and therefore longer t_d.

The plot show that the behaviour of the controller can actually be interpreted as a split of the docking into two behaviours. The evolution of such behaviour implies that the genetic algorithm first produces neural networks able to arrive to the final position without satisfying the orientation constraint, and then tuned those neural networks to be able to re-orient the spacecraft.

These results are preliminary and not yet final, as we still have certain discrepancies between the neurocontroller and the optimal control. Currently we try to improve the neurocontroller to practically replicate the optimal control results, as well as, trying to understand why evolution tries first to optimize position and then time. We also look into improving the genetic algorithm to reduce the time needed to generate good neurocontrollers.

7 Ongoing and Future Work

The preliminary results are interesting and encouraging for further studies. The results presented in the previous section show that evolved controllers produce trajectories fulfilling the boundary conditions given a fixed starting point. We are currently extending our research to address the robustness of the control and extend the generality by evolving a controller able to start from random positions as described at the beginning of the paper. Putting more emphasis on the behaviour obtained due to varying initial conditions and perturbations, we aim to generate

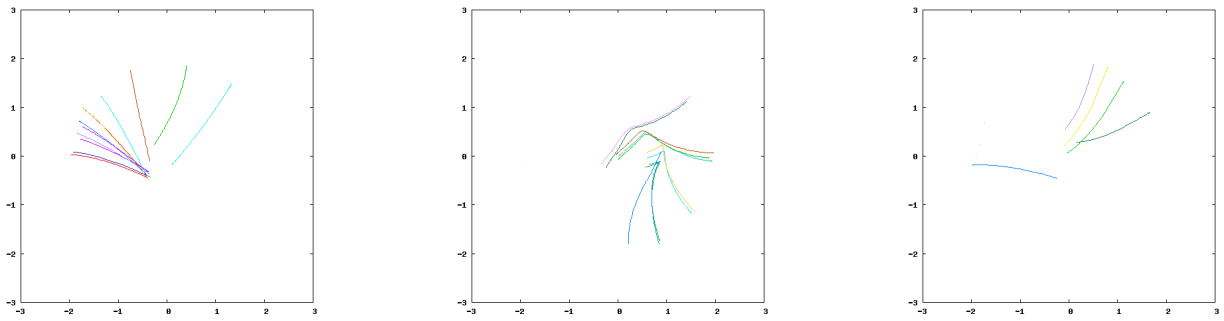

Figure 5. Three test runs showing preliminary results from neurocontrollers trained for multiple starting positions to enable a more robust and adaptive controller.

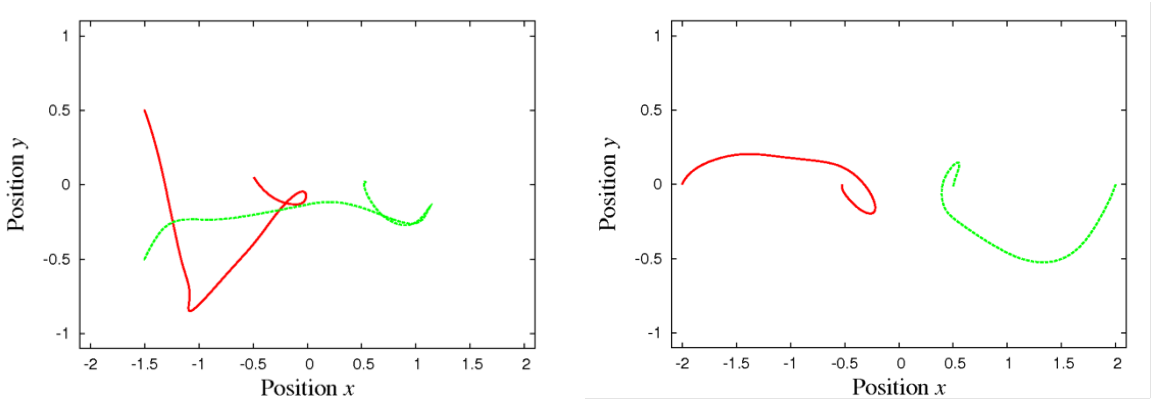

Figure 6. Trajectories generated for two spacecraft using the same neurocontroller.

more robust and fault-tolerant controllers.

Evolutions with varying initial positions and orientation are currently implemented. For these problems the first results, using the same approach as described above, were generated but are not yet conclusive[2]. A neurocontroller with one single hidden layer was used, but the number of hidden neurons was increased to 30. The tests are running simulations using 8 to 33 starting positions (with random attitude initialization) to enable generalization and a more robust controller. Each individual is evaluated for each starting position (with usually different initial conditions) and the fitness of the runs is averaged to define its fitness.

In Figure 5 we show some of the trajectories generated for multiple starting positions using the same neurocontroller consecutively. We can see that for most of the initial conditions the evolved ANNs are capable of steering the spacecraft towards the target location. However, the accuracy of the constraint satisfaction needs to be improved further to allow generalization.

Notice that our current approach relies on a very simple neurocontroller; the currently used ANN schema (Multi Layer Perceptron) cannot keep memory of the events it encounters. We are exploring the possibility of using dynamical neural networks, such as Continuous Time Recurrent Neural Networks (CTRNN), which are suggested to be better at handling system dynamics and at producing time-dependent behaviour.

Since the simulation is computationally expensive, future plans also include (i) parallelization using CUDA or MPI (i.e. increase of computational power), as well as, (ii) implementing the island model [9, 1] for faster evolution (i.e. reduction of iterations needed).

[2]The current progress can be found on the project webpage at `http://Juxi.net/projects/EvolvingDocking/`

7.1 Further Applications

Another experiment we are working on is the use of neurocontrollers for *forming patterns with multiple spacecraft*. Using the same dynamics and again an evolved neurocontroller, but here controlling two spacecraft, we have obtained promising results in role allocation and pattern forming. Notice that the control is homogeneous, that is, the same neural network, cloned in the two crafts, manages to control both of them in order to produce coordinated behaviour. In particular, the spacecraft are able to negotiate roles during a trial and assume a given configuration without a predefined position for each and without communication between the spacecraft. The fitness function is changed and this time punishes collisions while still rewarding time minimization and reduction of the distance between the agents and the desired final configuration. Some preliminary trajectories were calculated and are shown in Figure 6.

The different components of the fitness function aim to generate a collision-free deployment of the two agents from their initial position to one of the two possible final positions at the end of the simulation run. If the robot stays in close vicinity of the final target formation for more than 20 time steps, the trial is terminated and the fitness attributed to the neurocontroller is, as seen before, equal to $f = 1 + \frac{t_{max} - t_d}{t_{max}}$ (the same as described in Eq. 4).

These steps towards a single neurocontroller for multiple agents acting in space could also be quite useful for applications in on-orbit assembly of vast structures, a topic with an increasing interest within the space engineering community.

References

[1] C. Ampatzis, D. Izzo, M. Rucinski, and F. Biscani. Alife in the galapagos: migration effects on neurocontroller design. In *European Conference on Artificial Life*, 2009.

[2] C. Ampatzis, E. Tuci, V. Trianni, A. Christensen, and M. Dorigo. Evolving self-assembly in autonomous homogeneous robots: experiments with two physical robots. *Artificial Life*, 15(4):465–484, 2009.

[3] F. Biscani, D. Izzo, and C. Yam. A global optimisation toolbox for massively parallel engineering optimisation. In *International Conference on Astrodynamics Tools and Techniques*, 2010.

[4] B. Dachwald. Optimization of interplanetary solar sailcraft trajectories using evolutionary neurocontrol. *Journal of Guidance, Control, and Dynamics*, 27(1):66–72, 2004.

[5] Z. Denkowski, S. Migórski, and N. Papageorgiou. *An introduction to nonlinear analysis: applications*. Plenum Pub Corp, 2003.

[6] W. Fehse. *Automated rendezvous and docking of spacecraft*. Cambridge university press, 2003.

[7] C. Genshe and C. Xinhai. Improved fuzzy logic controller using genetic algorithm and its application to spacecraft rendezvous. In *IEEE Conference on Computer, Communication, Control and Power Engineering*, 1993.

[8] D. Izzo and L. Pettazzi. Autonomous and distributed motion planning for satellite swarm. *Journal of Guidance Control and Dynamics*, 30(2):449, 2007.

[9] D. Izzo, M. Rucinski, and C. Ampatzis. Parallel global optimisation meta-heuristics using an asynchronous island-model. In *IEEE Congress on Evolutionary Computation*, Trondheim, Norway, 2009.

[10] K. KrishnaKumar, S. Rickard, and S. Bartholomew. Adaptive neuro-control for spacecraft attitude control. *Neurocomputing*, 9(2):131–148, 1995.

[11] P. Nayak et al. Validating the DS-1 remote agent experiment. In *Artificial Intelligence, Robotics and Automation in Space*, 1999.

[12] S. Nolfi et al. How to evolve autonomous robots: Different approaches in evolutionary robotics, 1994.

[13] L. Pontryagin, V. Boltyanskii, R. Gamkrelidze, and E. Mishchenko. *The mathematical theory of optimal processes*. Interscience New York, 1962.

[14] R. Rojas. Was können neuronale Netze. *Mathematische Aspekte der angewandten Informatik*, pages 55–88, 1994. (in German).

[15] R. Sanner. Stable adaptive neurocontrollers for spacecraft and space robots. In *Goddard Conference on Space Applications of Artificial Intelligence and Emerging Information Technologies*, 1995.

[16] O. Von Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37(1):357–373, 1992.