UltraSwarm: A Further Step Towards a Flock of Miniature Helicopters

Renzo De Nardi and Owen Holland

Department of Computer Science
University of Essex
Colchester CO43SQ, United Kingdom
{rdenar, owen}@essex.ac.uk

Abstract. We describe further progress towards the development of a MAV (micro aerial vehicle) designed as an enabling tool to investigate aerial flocking. Our research focuses on the use of low cost off the shelf vehicles and sensors to enable fast prototyping and to reduce development costs. Details on the design of the embedded electronics and the modification of the chosen toy helicopter are presented, and the technique used for state estimation is described. The fusion of inertial data through an unscented Kalman filter is used to estimate the helicopter's state, and this forms the main input to the control system. Since no detailed dynamic model of the helicopter in use is available, a method is proposed for automated system identification, and for subsequent controller design based on artificial evolution. Preliminary results obtained with a dynamic simulator of a helicopter are reported, along with some encouraging results for tackling the problem of flocking.

1 Introduction

Swarm robotics is nowadays an established field of research; it offers the advantages of scalability, robustness through redundancy, flexibility, and reduced complexity of the individual robots. Within swarm intelligence, the topic of flocking deals with methods for controlling the motion of a group of agents (in real or virtual space) using rules directly inspired by ethological observations of real flocks of birds or schools of fish.

Since the seminal treatment of flocking developed by Reynolds [1] several researchers have explored the idea of flocking in real platforms or simulations. Most of the work involving flocking in real robots has concentrated on wheeled robots [2] [3] or airborne robots with limited dynamic capabilities [4]; due to the intrinsic dynamic and sensory limitations of the platforms used, none of these examples achieved really good-looking fluid flocking. Crowther addressed the problem of vehicles with more complex dynamics in his simulations of a flock of aircraft [5]. His research successfully demonstrated the potential usability of flocking as a decentralised traffic control method. In particular it showed that by simply changing the weights associated with Reynolds' rules, phase transitions appeared in the flock structure. However, omnidirectional perception was

assumed, and the presence of noise was neglected. Recently a development program carried out at the NASA Dryden Flight Research Center [6] demonstrated the coordination of two UAVs (in the form of two instrumented model aircraft) using Reynolds' flocking rules. GPS information was used to determine the relative positions of the aircraft, and this was sufficient to guarantee coordination. Unfortunately further details about this project are still unavailable. Another interesting application was developed by Atair Aerospace Inc. [7] in the domain of guided parafoils; a behaviour based algorithm inspired by flocking is used to ensure that all the payload-carrying parafoils will land together in the same area.

In the last few years, the problems of flocking and the distributed control of agents have gained popularity among the control system community [8][9][10]. The problems of stability, robustness, and the effects of sensing or communication delays are now being considered; see [11] for a more extensive review in the field. In a recent paper [12], Olfati Saber presents a theoretical approach to flocking; a single distance dependent potential function is defined to achieve both cohesion and separation. A particularly good definition of the potential function results in a smooth pairwise potential with a finite cut off that greatly simplifies the stability analysis. Since cohesion-separation and alignment can lead to fragmentation, an additional contribution to the control is added in the form of navigational feedback from progress towards a target point. The paper also presents an obstacle avoidance behaviour obtained by introducing fictitious agents near the obstacles.

Although the work of Olfati Saber addresses the problem of flocking using a simple point mass agent with double integrator dynamics, the results are supported by a sound theoretical analysis. It will be interesting to see if the same analysis can be extended to more dynamically complex vehicles. It is of course clear that extending this idea to highly nonlinear vehicles (e.g. helicopters) will constitute a big challenge in this respect. SamiloğluGazi et al. [13] give an example of how a simple physical constraint like a restriction on the turn angle may lead to oscillatory behaviours in the group.

2 The idea

As we have clearly seen in the introduction, achieving the flocking or swarming of real vehicles with complex dynamics is still an unsolved problem. Our work addresses many of the issues involved in this area: we aim to build a flock of dynamically complex vehicles (i.e. microhelicopters) to perform flocking in a real world scenario where the dynamics of the vehicles and the noisy outputs of the sensors are not negligible. The use of an aerial robotic platform removes the two dimensional limitation to which most of the previous research has been constrained, allowing for a scenario more similar to the one normally experienced by fish or birds.

In order to reduce development time and research costs, we aim to leverage as much as possible of the technology available in the market place - in other words, to take a COTS (commercial-off-the-shelf) approach. This translates into select-

ing a suitable commercially available vehicle (see section 3) and fitting it with the necessary off-the-shelf components; however, some hardware will inevitably have to be designed (see section 3.1).

These helicopters are of course structurally identical, but differences in the electric motors, the trim of the blades and the swashplate mechanism, and deformation of the very flexible foam blades will result in quantitatively different dynamic properties. The design of our controller should take this variability into account, together with the changes that will be induced by different sensor instrumentations of the same helicopter. These considerations mean that it will be more appropriate to develop a general method for automated model identification and controller design that can then be applied to different individual helicopters with different dynamics. A method based on machine learning techniques and artificial evolution is proposed in 4.

3 The helicopter platform

The ability to move in three dimensions is deemed to be an essential requirement of our system, as well as the need to be usable indoors (for ease of development). Only a few platforms can fulfil those two constraints: lighter than air vehicles (e.g. small blimps), and miniature helicopters. Small aircraft and slowflyers are clearly not an option, since our research arena (a cylinder of 12m diameter and 6m height) is too small to accommodate a flock of them. The more favourable size to payload ratio when compared to blimps led us to settle for a rotary wing solution.

Every roboticist and aircraft model enthusiast knows that the lift and hovering capability typical of a helicopter come at the cost of reduced dynamic stability. The helicopter flight controller is therefore a key element of our system. After evaluating several other models, a helicopter with a counter rotating dual rotor configuration was chosen for this study [14] (see figure 1). The counter rotating configuration is well known for delivering high efficiency as well as achieving excellent stability thanks to the direct compensation of the torque between the two rotors. The model we have selected uses a conventional fully controlled lower rotor, and an upper rotor fitted with a 45 degree stabilising bar. The stabilising bar exploits gyroscopic forces in order to counteract sudden changes in shaft inclination. This results in improved stability, but of course this comes at the cost of a reduced response to control commands. Testing by human pilots showed the model to be more stable and easier to fly than conventional single rotor helicopters; although not suitable for advanced aerobatic manoeuvres, the helicopter retains the manoeuvrability necessary to perform flocking.

Testing also established that the model can be flown with a payload of about 40g for about 15 minutes; we deem this sufficient to achieve the project's aims.

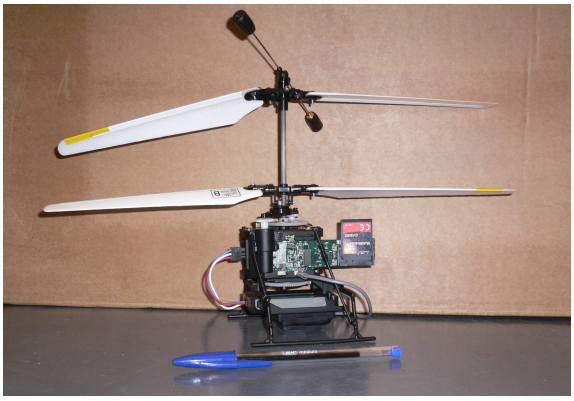

Fig. 1. Helicopter retrofitted with the new electronics, the Gumstix computer and the IMU.

3.1 Electronics and sensors

Since the helicopter is sold as a remotely controlled toy, the first step towards autonomy involved the complete redesign and replacement of the helicopter's electronics.

The bulk of the work on the platform involved the design and manufacture of an electronic board that interfaces a Gumstix SBC (single board computer) to the two main electric motors powering the rotors, and to the servos controlling the swashplate. The board was manufactured in surface mount technology, and is based around a low power 40 MHz ARM7 microcontroller. The microcontroller offers two serial ports for reflashing and for communication with the Gumstix, two i2c ports to interface to the ultrasonic sensors and the IMU (inertial measurement unit), and four PWM outputs to drive motors and servos. External interrupt inputs are also available to interface to two rotor speed encoders; an additional analogue input permits us to monitor the battery voltage. A second companion electronic board accommodates the power stabilisation circuitry and the highly efficient MOSFET motor drivers. The two electronic boards were specifically designed to fit within the original central housing of the helicopter in order to maintain all the moments of inertia as close as possible to those of the original helicopter.

The choice of having an additional low level microcontroller that directly interacts with the hardware was made to guarantee a high degree of reliability for the control system. The absence of an operating system allows a tighter coupling with the hardware, offering real-time execution of the critical code needed for helicopter stabilisation. The processing power of the Gumstix SBC will be entirely dedicated to the high level software (e.g. guidance and communication), where the use of the Linux operating system will allow for easy and fast development. Along with the electronics, a set of low level routines has been developed

to allow the microcontroller to interact with the hardware and enable communication with the Gumstix SBC. Thanks to the Bluetooth wireless communication present on the Gumstix we are now at a stage in which it is possible to command all the helicopter flight controls from a remote computer. A simple yaw stabilization based on the IMU gyros has been implemented to aid during manual flight; being able to fly the helicopter manually by exploiting the Bluetooth connection is obviously important to test the helicopter hardware, but will be crucial during the process of data collection for modelling purposes. Previous work of the authors [15] with a similar helicopter and electronics showed how the delay introduced by the Bluetooth connection still allows for the full control of the flight machine. In the final system the communication delay will not be an issue since the control algorithm will run on board.

Ultrasonic sensors and an IMU are the crucial sensors for the control and stabilisation of the vehicle; however, for flocking, each helicopter also needs to determine the range and bearing of its nearby flockmates. Fortunately our indoor arena will shortly be provided with a state of the art 3D tracking system based on infrared markers that will be able to determine with great accuracy (i.e. to within a few millimetres) the position of each helicopter. The relative positions computed by a stationary computer can then be fed back to each of the helicopters through the wireless data link. This will enable us to test the basic flocking algorithm. Further research will investigate the direct sensing of relative position using RSSI (received-signal-strength-indication) from the communication channels, the use of onboard radio beacons, and also onboard vision.

3.2 State estimation

A key problem is the need for the real-time computation of state estimation. This includes the helicopter's attitude (e.g. the Eulerian angles with reference to an Earth frame ϕ, θ, ψ), its rotational speeds (p, q, r), and its linear velocity and position (u, v, w and x, y, z). Attitudes and rotational speeds are needed for helicopter stabilisation; the position in the form of the relative distances between the members of the flock will be used for regulating flocking.

Given the reduced payload only a very light IMU can be carried on board. We have selected the Memsense nIMU [16] which includes three linear accelerometers, three gyros, and three magnetic field sensors. MEMS (Micro-Electro-Mechanical Systems) technology allows the production of inertial sensors of very compact size and reduced weight (the whole IMU has a weight of only 15g); however, their performance in terms of error and temperature bias tends to be significantly worse than alternative navigation grade solutions using optical methods. The dependency on temperature variation is already compensated within the IMU sensor, but the effects of noise and, more importantly, of the drift that affects gyros and accelerometers must still be compensated to allow sufficiently accurate data to be obtained.

The main inputs used for the state estimation are the measured rotational speed and linear acceleration; both are affected by noise, and by a time-variant bias. The magnetometer readings, the ultrasonic sensor values, and possibly the

position obtained from the tracking system will be used to "correct" the inertial data. The magnetometers and ultrasonic sensors are not affected by time variant bias, and so have error characteristics complementary to those of the inertial sensor; the fusion of the two types of sensor data will allow us to improve the state estimation.

The most commonly used techniques for data fusion rely on Bayesian filtering techniques, among which the best known is probably the EKF (extended Kalman filter). Although it has been successfully applied to helicopter state estimation problems ([17] [18]) the EKF has some weaknesses when compared to other similar approaches. Van der Merwe and Wan conducted a comparison analysis between an EKF and a UKF (unscented Kalman filter) The analysis [19] shows that since the sensor model used in the filter is strongly nonlinear (due to the change of coordinates), the UKF can improve the estimation performance. In addition the implementation of the UKF is comparatively much simpler than that of the EKF since the there is no need to calculate the derivatives of the state equations. Given our limited computational power, it will also be interesting to explore the possibility of implementing the UKF in its square root form, [20] which presents improved numerical stability along with reduced computational complexity.

In order to limit the computational complexity, our system model is simply that of a 6DoF rigid body freely moving in a 3D space. Position, speeds, Euler angle and sensor bias constitute the state estimated by the filter. The system equations are represented by the classic equation of motion of a rigid body in a 3D space. The acceleration and rotational speed values coming from the IMU are used as control inputs to the model in order to propagate the system state. Readings from the ultrasonics sensors and the magnetometers constitute the measurements that will allow the correction, through the observation model, of the predicted state in the interactive prediction-correction fashion typical of Bayesian filtering.

The system model includes the update equations necessary to estimate the biases of the accelerometers and gyros, and so it will therefore allow them to be compensated.

4 An automated design method

We already mentioned in section 2 the clear advantages offered by using an automated method to deal with the unknown dynamic differences between the aerial vehicles. Such an automated method will be useful in the future as well, as we plan to move the system outdoors, and therefore to use heavier and more complex helicopters.

Because of its complexity and severe nonlinearity, the understanding of helicopter aerodynamics is still relatively poor, and the direct estimation of model parameters from experimental flight data still remains the only effective way to accurately capture the dynamics of a vehicle with unknown characteristics.

4.1 Model identification

The most common approach to model identification for small helicopters is simply a specialisation of the classical approach widely used for full scale vehicles. Such models are based on a knowledge of aerodynamic principles, and the derived equations account for the lift and sideforce generated by the rotors, the effects of drag on the fuselage, the inertia and other effects of the stabilising bar, and the coupling between the major axes of the helicopter. This typically yields a very complicated model expressed in helicopter body coordinates, and with several tens of parameters. From a linearisation of this type of model, a state model can be derived; its free parameters can then be learned from flight data by using data association techniques in the frequency domain. By making provision for the selection of meaningful flight data, and by enabling the possibility of including several equilibrium points in the flight envelope, a sufficiently accurate helicopter model can be produced. Mettler *et al.* [21] and La Civita *et al.* [22] describe two successful applications of this technique.

A general weakness can be attributed to this method; since the model is ultimately linear, effects like inertia and gravity which involve the nonlinear contributions of velocity and angular rate are really hard to capture. To finesse this problem, Abbeel *et al.* [23] proposed an alternative approach to modelling based on acceleration prediction.

Physics tells us that the relationships between effects like inertia, gravity and acceleration are often straightforward, and so it is clear then that a model based on acceleration prediction could possibly be both simple and effective in describing these effects. Such a model will be expressed in acceleration terms, and so, in order to obtain the state vector, we will need to integrate the acceleration contributions at every timestep. Since the accelerations are expressed in body coordinates, and since the body reference frame changes at every timestep, a change of coordinates must be performed at every timestep before proceeding with the computation. Here we see that since the algorithm explicitly takes care of these changes of coordinates, the learning is greatly simplified since this highly nonlinear functional relationship does not need to be learned. (However, the model resulting from the integration of the acceleration prediction is still of course nonlinear).

The general model proposed by Abbeel represents a 6 DoF vehicle and deliberately avoids giving any aerodynamic meaning to the dynamic equations; the only system knowledge introduced is reflected in the choice of body-centred coordinates, and of the axes of symmetry. According to the authors, its generality means that the same model could also be used with minor changes for vehicles very different from a helicopter. Faced with the specific situation of learning a model of the helicopter used in our project, we can therefore see an excellent opportunity for the use of the technique proposed by Abbeel, since only a very limited knowledge of the vehicle is necessary.

It is necessary to point out that this type of model, although potentially very accurate, still has 15 parameters and is fairly computationally expensive. This is not a problem when we are evolving the controller off-line, but unfortunately

it prevents us from using it real time in association with our Bayesian filtering algorithm. For this reason the simpler 6DoF model explained in 3.2 is used in the UKF.

4.2 Controller design

The evolutionary design of controllers based on neural networks has proven to be an effective methodology for designing controllers for simple robotics problems [24], and also for agents in computer games [25] [26]. A controller based on neural networks with specific topologies has also been applied successfully to the domain of helicopter control [27] [28], although in this case the training was based on reinforcement learning. Building on this previous work we have developed a method for training a custom designed neural network using a form of neuroevolution [29].

The controllers (i.e. the neural networks) have a fixed topology and a fixed *tanh* activation function, and so a simple fixed length array of real numbers is sufficient to represent the genome of each of the controllers. A modular topology (see figure 2) inspired by the layout of a multiloop PID controller was chosen since it had been found to greatly improve evolvability .

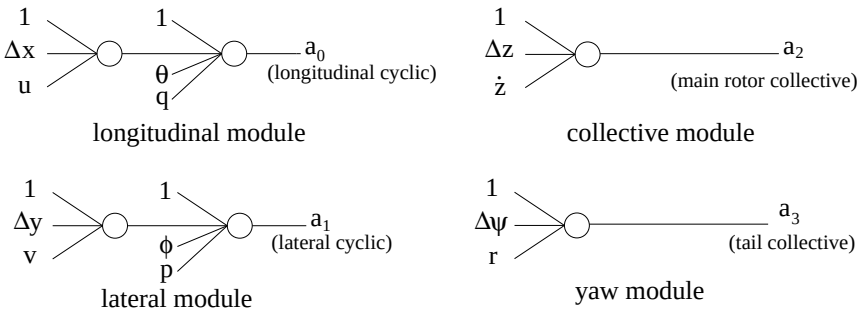

Fig. 2. Modular network used in the waypoint task. The same network topology was also used for the velocity task but the inputs $\Delta x, \Delta y$ ans Δz were substituted respectively by $\Delta u, \Delta v$ ans $\Delta \dot{z}$

A variation of ES (Evolutionary Strategies), with a total population of 33 individuals and an elite of 10, is used to evolve the weights. The first population consists of neural networks with small random synaptic weights; each controller is then evaluated on the task at hand. This involves using the controller under test to fly the simulated helicopter model and try to achieve the desired task (e.g. flying a set of waypoints, or reaching a specific vectorial speed). Its fitness represents the ability demonstrated by the controller on the specific task used. The population is then sorted by fitness and the worst 23 individuals are replaced with mutated versions of the 10 best individuals (the elite). The algorithm does

not apply recombination; the weights of the network are simply mutated by adding a random value (drawn from a Gaussian distribution with mean 0 and standard deviation 0.01). In this way, a new population is created, and the evolutionary process can then be repeated for the next generation.

The possibility of defining different tasks and the different fitness functions associated with them allows us to customise the helicopter controller to our needs. This constitutes a really valuable option, and offers clear advantages when compared to the traditional manual design of the controller.

As noted in section 3.2 our algorithms for state estimation and model identification have not yet been validated; this is due to a manufacturing problem with the IMU sensor that is currently being rectified. In the meantime, a freely available dynamic helicopter simulator with dynamics qualitatively similar to our helicopter [30] was used to test our approach to the evolution of the controller. This simulator accurately reproduces the dynamics of the XCell 60 model helicopter. Blade element theory is used as the basis for the computation of rotor thrust and drag forces, and the main rotor dynamics and stabilising bar are modelled as proposed in Mettler *et al.* [21]. Dynamic coupling and aerodynamics effects are also modelled. The simulator outputs the same state variables as will be available from the Hirobo helicopter, and accepts the same flight control inputs. Although qualitatively similar to the Hirobo in all essential respects, the simulated helicopter is much less stable[1], and so it definitely constitutes a challenging test bench for our design approach.

Two different tasks were devised to test the design method. In the first, the helicopter is commanded to perform a specific flight trajectory; this controller will be useful for testing autonomous flight. In the second, the controller is requested to fly the helicopter with a specific (vectorial) velocity; this controller gives a basis on top of which the classic Reynolds flocking rules could be applied.

In both tasks, there is an initial stage in which the controller is evolved for a few tens of generations using the heading error as the only fitness function. This evolution is very quick, and produces a minimal yaw stabilisation that enables the system to "bootstrap" the subsequent task evolution.

In the first task the controller is required to fly the helicopter along a predefined random generated set of waypoints. The fitness is based on progress along the path so defined; a waypoint was deemed to be visited when the centre of the helicopter approached within 1 foot of it. The waypoints were placed at a mean distance of 17.5 feet from each other. To encourage a straight path between the waypoint the fitness was reduced if the helicopter deviated from the path. Additional penalties were awarded for differences from the correct altitude and heading. The complete fitness function is this:

$$f = \frac{\sum_{i=0}^{N} (w_h P_{chain} \mid z - z_{next} \mid - \mid \psi - \psi_{ref} \mid)}{N}. \quad (1)$$

[1] One of the authors, who can fly the Hirobo helicopter very competently, has consistently failed to control the simulated model for more than a few seconds of simulated time.

Where N is the number of timesteps allowed to execute the task (for evolution $N = 1000$ was used) and z_{next} , ψ_{ref} are respectively the altitude of the next waypoint, and the fixed reference heading. The factor w_h is equal to one if the helicopter is on the shortest path between waypoints and decays as the cube of the orthogonal distance from it.

The input to the network are formed by the helicopter attitudes ϕ, θ, ψ, the rotational speeds p, q, r, the linear speeds u, v, w, and the relative distance to the next waypoint $\Delta x, \Delta y, \Delta z$ (both speeds are expressed in the helicopter body reference frame).

A sample of the trajectory flown by the best controller during a test run is shown in figure 3. As we can see, the evolved controller exhibits the ability to fly correctly through the predesigned waypoint chain. Regardless of the relative distance or the position between the waypoints, the path is very smooth.

A second task was devised with the Reynolds flocking rules in mind. At the beginning of the task the helicopter is started in a hovering position, and a randomly generated increment in velocity (expressed in the three helicopter frame of reference components) is requested. Each single velocity increment can have a value in the range $[-3.5 \div 3.5] ft/s$; if the sum of the current velocity and of the increment exceeds $10 ft/s$ a cut-off is applied to guarantee a requested speed consistent with the helicopter's capabilities. Along with the requested change in velocity, the required duration of the change is also specified. This is also a random value in the range $[12 \div 350]$ timesteps. The fitness is simply the squared magnitude of the error between the commanded and the instantaneous helicopter velocity:

$$f = \frac{\sum_{i=0}^{N} \|v(t) - v_c(t)\|_2^2}{N}. \tag{2}$$

Where N is the number of timesteps allowed to execute the task (for evolution $N = 1000$ was used), $v(t)$ is the velocity of the helicopter in the body frame coordinates at time t and $v_c(t)$ is the velocity commanded at the same time instant.

The inputs to the network are the helicopter attitudes ϕ, θ, ψ and rotational speeds p, q, r, the linear speeds in the body reference frame u, v, v_z, and the difference between the commanded speeds and the actual speeds $\Delta u, \Delta v, \Delta v_z$.

Sample plots of the difference between the commanded value of the velocity and the actual velocity are displayed in figure 4. It is clear that the helicopter speed varies in accordance with the request; unsurprisingly, a steady state error is present. It is noteworthy that the responses to the input steps do not show any signs of instability, and that a clear coupling between the longitudinal and lateral speeds is present. Future work will add complexity in the network structure to attempt to compensate for the coupling.

In the course of obtaining the results just described, various network topologies and incremental evolution approaches were investigated (for details see [29]). Without incorporating some domain knowledge into the evolutionary process, we were unable to evolve successful controllers. Domain knowledge was introduced in the form of the network topology, which neglects any coupling between the

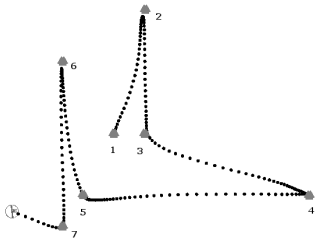

Fig. 3. Trajectory of a modular network controller after completing 1100 timesteps of the waypoint task.

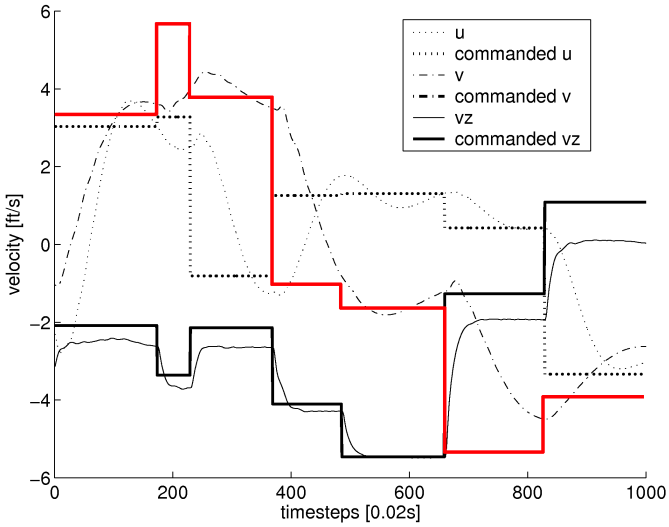

Fig. 4. Plot of the commanded speed vs the real helicopter speed for the best controller evolved with the velocity task.

lateral, longitudinal, and vertical axes. Evolving the yaw controller first was also crucial for the evolutionary process, confirming the findings of other researchers about the nature and benefits of incremental evolution.

5 Future work

In the immediate future we will implement and evaluate the approach of data collection, state estimation, system identification, and controller design on our model helicopter.

The first step will concentrate on the validation of the unscented Kalman filtering approach; the maximum update frequency and also the numerical robustness need to be determined. The performance of the filter algorithm in terms of noise and drift also needs to be tested to ensure that the data will be adequate for control.

Model identification based on recorded flight data will constitute the next step. By its very nature the system identification technique will provide us with a quantitative estimation of the error between the simulated and real trajectory. We expect that the simulator will not be able to predict the trajectory of the real helicopter for more then a short period of time, due to the accumulation of error. However, the dynamic response of the model to the control input, which is what is needed to evolve a controller, will always resemble that of the real helicopter.

Artificial evolution will then be applied to produce controllers tailored to our helicopter. Several controllers chosen from those with good fitness will than be evaluated directly on the real helicopter.

Finally, a controller will be implemented on board the helicopter and the sensor to motor action loop will be closed, allowing us to test autonomous flight.

The work will then proceed with the investigation of strategies for achieving flocking; these will initially be based on the classical rules of cohesion, separation, and velocity matching.

6 Concluding remarks

The work presented here is clearly still in its early stages, but is following a clear path supported by existing research findings. The results achieved in the simulation and testing carried out so far are encouraging; we recognise however that porting the results obtained in simulation to a real system is very often problematical.

References

1. Reynolds, C.: Flocks, herds, and schools: A distributed behavioral model. In: Proceedings of the Conference on Computer Graphics (SIGGRAPH). Volume 21:4. (1987) 25–34
2. Mataric, M.: Interaction and intelligent behavior. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA (1995)
3. Kelly, I., Keating, D.: Flocking by the fusion of sonar and active infrared sensors on physical autonomous mobile robots. In: Proceedings of The Third Int. Conf. on Mechatronics and Machine Vision in Practice. Volume 1. (1996) 1–4
4. Welsby, J., Melhuish, C.: Autonomous minimalist following in three dimensions: A study with small-scale dirigibles. In: Proceedings of Towards Intelligent Mobile Robots Manchster. (2001)
5. Crowther, B., Riviere, X.: Flocking of autonomous unmanned air vehicles. In: Proceeding of the 17^{th} UAV System conference, Bristol UK. (2002)
6. NASA, D.F.R.C.: New flight software allows UAVs to team up for virtual fire experiment. (http://www.nasa.gov/centers/dryden/news/NewsReleases/2005/05-12.html)
7. Calise, A., Preston, D.: Swarming/flocking and collision avoidance for mass airdrop of autonomous guided parafoils. In: AIAA Guidance, Navigation, and Control Conference and Exhibit. (2005)
8. Jadbabaie, A., Lin, J., Morse, A.: Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Trans. Automatic Control **48**(6) (2003) 998–1001
9. Tanner, H., Jadbabaie, A., Pappas, G.: Stable flocking of mobile agents. part I: Static topology. In: Proceedings of the 42nd IEEE Conference on Decision and Control. (2003) 2010–2015
10. Tanner, H., Jadbabaie, A., Pappas, G.: Stable flocking of mobile agents. part II: Dynamic topology. In: Proceedings of the 42nd IEEE Conference on Decision and Control. (2003) 2016–2021

11. Gazi, V., Fidan, B.: Review of control and coordination of multi-agent dynamic systems: models and approaches. In Sahin, E., Spears, W., Winfield, A., eds.: Swarm Robotics Workshop (SAB06). Lecture Notes in Computer Science (2006)
12. Olfati-Saber, R.: Flocking for multi-agent dynamic systems: Algorithms and theory. IEEE Transaction on Automatic Control **51**(3) (March 2006)
13. Samiloğlu, a., Gazi, V., Koku, B.: An empirical study on the motion of self-propelled particles with turn angle restrictions. In Sahin, E., Spears, W., Winfield, A., eds.: Swarm Robotics Workshop (SAB06). Lecture Notes in Computer Science (2006)
14. Hirobo Limited: XRB Lama helicopter. (http://model.hirobo.co.jp/products/0301-905/index.html)
15. Holland, O.E., Woods, J., De Nardi, R., Clark, A.: Beyond swarm intelligence: The UltraSwarm. In: Proceedings of the IEEE Swarm Intelligence Symposium (SIS2005), IEEE (2005)
16. Memsense: nIMU nano inertial measurement unit. (http://www.memsense.com/content/products/Datasheets/nIMUv1_92.pdf)
17. Jun, M., Roumeliotis, S., G.S., S.: State estimation via sensor modeling for helicopter control using an indirect kalman filter. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. (1999)
18. Gavrilets, V.: Autonomous Aerobatic Manouvering of Miniature. PhD thesis, Massachusetts Institute of Technology (2003)
19. van der Merwe, R., Wan, E.A.: Sigma-point kalman filters for integrated navigation. In: 60th Annual Meeting of The Institute of Navigation (ION). (2004)
20. van der Merwe, R., Wan, E.A.: The square-root unscented kalman filter for state and parameter-estimation. In: International Conference on Acoustics, Speech, and Signal Processing. (2001)
21. Mettler, B., Tischler, M., Kanade, T.: System identification of a model-scale helicopter. Technical Report CMU-RI-TR-00-03, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (2000)
22. La Civita, M., Messner, W.C., Kanade, T.: Modeling of small-scale helicopters with integrated first-principles and system-identification techniques. In: American helicopter society 58th annual forum. (2002)
23. Abbeel, P., Ganapathi, V., Ng, A.Y.: Modeling vehicular dynamics, with application to modeling helicopters. In: Neural Information Processing Systems. (2005)
24. Nolfi, S., Floreano, D.: Evolutionary robotics. MIT Press, Cambridge, MA (2000)
25. Togelius, J., Lucas, S.M.: Evolving controllers for simulated car racing. In: Proceedings of the Congress on Evolutionary Computation. (2005)
26. Togelius, J., Lucas, S.M.: Forcing neurocontrollers to exploit sensory symmetry through hard-wired modularity in the game of cellz. In: Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games CIG05. (2005) 37–43
27. Ng, A., Kim, H., Jordan, M., Sastry, S., Ballianda, S.: Autonomous helicopter flight via reinforcement learning. Advances in Neural Information Processing Systems (2004)
28. Ng, A., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., Liang, E.: Autonomous inverted helicopter flight via reinforcement learning. In: Proceedings of the International Symposium on Experimental Robotics. (2004)
29. De Nardi, R., Togelius, J., Holland, O., Lucas, S.: Neural networks for helicopter control: Why modularity matters. In: IEEE Congress on Evolutionary Computation. (2006)
30. Autopilot: Do it yourself UAV. (http://autopilot.sourceforge.net)