An Approach to Fully Automatic Aircraft Collision Avoidance and Navigation

SANTIAGO ÁLVAREZ DE TOLEDO, JOSÉ M. BARREIRO, JOSÉ L. FUERTES, ÁNGEL L. GONZÁLEZ AND JUAN A. LARA
Department of Artificial Intelligence
School of Computing, Technical University of Madrid
Campus de Montegancedo, 28660, Boadilla del Monte, Madrid
SPAIN
http://www.dia.fi.upm.es

Abstract: - The main goal of this paper is to show a way to increase air traffic safety through automation and this way permit its growth. An algorithm has been built to allow fully automatic mid-air collision, ground, terrain or building avoidance, without any intervention from the pilot. Additionally, the algorithm allows fully automatic navigation. These capabilities have been tested through simulation. We plan to use this software with a flight simulator. The software will take the pilot's place in these tests. This way the software will be evaluated in a similar way to how pilots learn and are assessed.

Key-Words: - Artificial Intelligence, Machine Learning, Collision Avoidance, Reinforcement Learning, Navigation, Air Traffic Control.

1 Introduction

World air traffic is permanently on the increase. Air traffic density is growing and will continue to do so in areas of high population density and rising economic level. This traffic density growth raises constant concern over air traffic conflicts and collisions [1] and safety generally.

Communication between the controller and pilots, originally by voice only in a language, which is sometimes neither the pilot's nor the controller's mother tongue, is a source of ambiguities and misunderstandings. Some such misunderstandings still cause safety problems today and not only involve small airplanes but also big commercial aircraft.
Generally, automation aims to do away with the "human errors" that unfortunately still occur [2]. Improvements like cockpit printers and displays and communication via *Datalink* moved in this direction [3].

Most input information is now processed automatically in today's large commercial aircraft. First, this is necessary when the pilot does not have a clear view of the area. Second, it is safer then leaving it to the pilot's subjective assessment and judgment. In this way, GPS, Traffic Collision Avoidance Systems (TCAS II or T2CAS) [4], ground and airport databases and systems (Enhanced Ground Proximity Warning System – EGPWS) [5] and so on automatically report the aircraft's position, the position and trajectory of a nearby aircraft, the surrounding terrain, airport situation, buildings, even aerials. These systems automatically calculate whether a collision with another aircraft is possible, a flight into terrain is hazardous, or the glideslope descent rate is excessive.

However, within this continuous trend towards automation, the automatic action or maneuver sometimes needs an as yet unattained level of intelligence and flexibility. The input systems calculate and clearly explain what the pilot should do but not exactly how he or she should do it.

Finally, there are some aeronautical problems that could benefit from the software presented in this paper. One is the need for faster reaction time for collision avoidance in military aircraft. Another is the need for a safer, possibly automatic, way to keep an aircraft hijack situation under control.

2 New Algorithm

The algorithm is based on reinforcement learning [6]. This way the agent modifies its action as it proceeds in the learning process. The agent first acts unintelligently with a very high error rate. Later, after taking a near correct action in a specific circumstance, the agent gathers information on this action, as well as on the erroneous actions and their circumstances. The agent tries to repeat its successes and avoid its mistakes in the same or similar circumstances. The agent optimizes its performance by relating the best actions to the same or similar environments and circumstances and also by relating actions to be avoided to the same or similar circumstances [7].

The algorithm provides specific learning principles that specially support and speed up the learning process:

Mistakes and successes: provisional and final results. If we aim to build a smart powerful and flexible algorithm, the information on the results has to give both positive and negative in-process feedback for each action and not wait for the end result.

Granularity: intensity of resulting values. One way of enriching the resulting data is to have the system know the intensity or extent of the resulting values: how positive or negative the successful or mistaken results were, as opposed to just the YES or NO feedback [8].

Measurement of time. When there is a time lapse between the action and the result, even if both happen repeatedly, there are other intermediate results that could be candidate effects for the combined cause (environment plus action). Taking into account these time lapses helps to establish associations that would not be detected if only the simultaneous occurrence of a perception, an action and a result were considered. Other time lapses between the environmental data, the action and the perceived result are also taken into account. Measuring such time lapses facilitates the statistical analysis and speeds up the learning association.

Back propagation of the results evaluation. The perceptions that do not have a positive or negative value of their own, but are simultaneous or contiguous to others that do, receive a corresponding back-propagated positive or negative value.

Multisensorial patterns. When groups of different and heterogeneous sensors are used, the input is richer and much more easily identified, and the action is therefore better selected.

Contiguous patterns. When there is a lapse of time between events its elements may not be alike or even similar. A pattern comprising all these contiguous perceptions enables the identification of this reality. Additionally, there is the possibility of juggling with more actions, since there are more input combinations within one string. Each combination is potentially associated with a different action.

Direct non-learnt action. In the early stages, before the agent has learnt enough to survive, it should take some direct intelligent actions that would be triggered by some specific perceptions or stimuli [9].

Action probability. Actions should be taken in proportion to their probability of bringing about a successful result. The algorithm also includes a proportion of exploratory actions.

Deletion of non-meaningful information. When millions of perceptions, actions and result patterns are being processed, many associations are not meaningful and therefore should be deleted. In some cases, where there are huge amounts of data, most associations are further deleted.

The algorithm has five main steps:

Step 1. A number of sensors perceive the environment to generate the above multisensorial perception pattern. This perception pattern is evaluated according to other positive and negative value sensors to output a mean value.

Step 2. An action is selected in the light of all the historical information about the positive and negative values associated with the perception-action pairs. This is done according to the probabilities of a more positive or less negative value being generated for the pairs. Nevertheless, exploration, i.e. selection of a different action than the one dictated by experience, takes place in the early stages to increase the experience level. Some direct actions, e.g. special action routines for getting out of difficult situations during the early learning period, when there is not enough accumulated experience, are also taken in a few exceptional circumstances.

Step 3. The perception pattern and action pattern pairs are associated with the resulting value output during the action time cycle or later. This resulting value is associated with a decreasing weight depending on the number of cycles in which the result occurred versus the perception and versus the action, according to formula

$$V(p_k, a_j)_i = V(p_k, a_j)_{i-1} + \alpha_{pa}^{k} \alpha_{av}^{j} v \,, \qquad \textbf{(1)}$$

where V is the value of the pair association, p is the perception pattern, a is the action pattern, I is the cycle in which the resulting value occurs and α is the discount value per cycle that is applied to reduce the resulting value to the historical value (the above measurement of time).

Step 4. Additionally, each perception pattern has a unique value that depends on the resulting value perceived later, its intensity and sign, and the number of cycles between the result and the perception. At the same time, the less certainty there is of this value being associated with the patterns, the bigger the part of the resulting value back-propagated towards previous perception patterns is. The formula for this is

$$V_{P_i} = \frac{c_{P_{i-1}} V_{P_{i-1}} + \frac{\alpha v}{2^j}}{c_{P_{i-1}} + \frac{\alpha v}{2^j}} \qquad (2)$$

where V_{Pi} is the value of the perception pattern in the present cycle, αv is the discounted value depending on the elapsed j cycles, c_{Pi-1} is the certainty of the perception pattern value in the previous cycle, V_{Pi-1} is the value of the perception pattern in the previous cycle.

The above certainty of such a value for the perception pattern depends on the actual resulting value and the number of cycles between the result and the perception (measurement of time). The formula for this is

$$c_{pi} = c_{P_{i-1}} + \frac{\alpha v}{2^j}, \qquad (3)$$

where c_{pi} is the certainty.

Step 5 of the algorithm is the deletion phase, when the superfluous information is deleted. This step deletes the perception patterns that are less frequent than a specific certainty factor, as well as the pair associations with a lower level of association than another constant. If they are higher than those factors, they are reduced by a variable amount.

3 Experimentation and Testing

The software, and algorithm, is highly adaptable to dynamic 2D or 3D environments, giving agent movements, collision avoidance, obstacle contouring and navigation a great deal of flexibility. It has proven therefore to be very useful for aeronautics, submarines, ground vehicles and robotics. The algorithm has been generalized for all kinds of input or output patterns. Therefore, it can be used in the aeronautics application described here with different input (environmental) and output (maneuvers) data or the same data in a wide variety of different formats. Owing to the algorithm's generalization, it is also adaptable to different dynamic applications as mentioned above.

A simulator has been built using the above algorithm, providing simulated input information similar to what is today relayed to the aircraft and feeds the Flight Management System (FMS) and other on-board systems. Input data is aircraft, airport and start of the runway positions, nearby terrain, building and obstacle positions and so on. Output information is forward maneuvers or change the trajectory up or down or turning in a specific direction.

The simulation starts with a learning phase through which the agent/aircraft proceeds, continuously improving its dynamic behavior. When the learning curve has converged the software is ready for use.

A first experiment was run simulating an aircraft near the airport runway. To be able to clearly track the learning process, an ultralight or very small aircraft was considered. As this kind of aircraft can approach the airport laterally and does not necessarily have to align with the start of the runway until the end of the glideslope, there is a large space with the shape of a virtual cone in which the aircraft can decide the landing approach. The cone was designed with virtual walls which define the maximum permitted altitude and lateral angles for approach. The aircraft can physically go through the cone's virtual walls, but the landing maneuver will not be correct unless it keeps inside. The start of the runway is at the end of this virtual cone. In a such large space, all possible errors of an unskilled / pre-learning agent or aircraft are crystal clear since the ultralight or small aircraft initially has no maneuvering limitations, and the learning process is easier to follow. The glideslope of a large aircraft would have been like a narrow tunnel.

In this experiment there are a high number of patterns. These patterns depend on the aircraft's position, orientation to the start of the runway and perception of the virtual cone walls. These complex patterns are multiplied combinatorially to produce hundreds of thousands. As an added difficulty, the agent/aircraft starts every trial from a different and randomly selected point and with a different and random orientation with respect to the start of the runway. It never knows where it is in the space (this is not necessary in this experiment). All the information that the agent/aircraft has is its orientation towards the runway and its proximity to the virtual cone walls –and this only when it is very close.

First comes the simulated learning phase. In the first trial, the aircraft makes a number of senseless movements. No limitations have been placed on these movements. Therefore, the number of possible actions is higher and mistaken actions are more likely. This makes learning more difficult, but more obvious and easy to follow when it takes place. If this were not a simulation, the first movements of the aircraft would send it crashing straight to the ground. When the agent starts to navigate with some equilibrium, we find that it does not move in any particular direction in a sort of fly anywhere navigation. Later it starts to head towards the start of the runway, but reaches the end of the glideslope dangerously. This would in practice force it to abort the landing. Finally, the agent learns to navigate towards the runway, to keep to the right direction, to correct its trajectory when it is likely to move outside the virtual cone (either laterally or vertically) and to safely end at the start of the runway. The shape of the virtual cone, even if it is a narrow tunnel as in the case of a big aircraft, has no impact on the positive results (Figure 1)

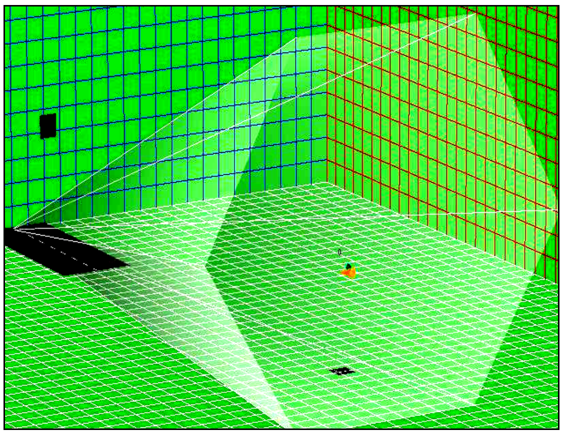
Fig. 1. Aircraft with vertical and horizontally produced shadows; runway.

After the learning process, the successful landing levels are practically 100% and the optimum trajectory levels are 98%. The first successful landings take place after some 1,000 trials. Normally there are about 60,000 trials, each with up to 600 cycles or independent actions/movement (Figure 2).

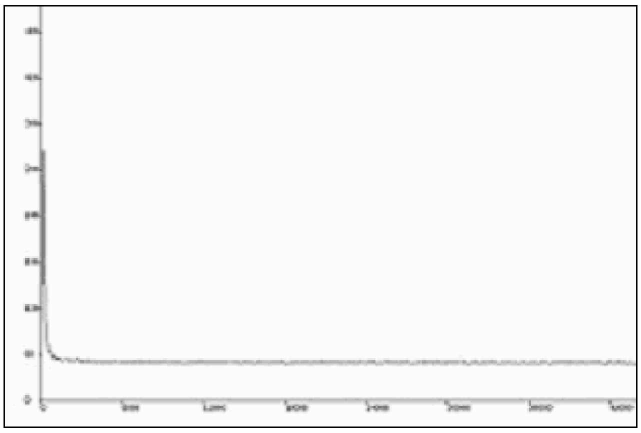
Fig. 2: Learning curve with very early convergence

Other experiments have been run with the agent avoiding flight into terrain or a hazardous ground proximity and a number of other obstacles (buildings and so on). One principle of the algorithm is that the proximity to physical objects (aircraft, terrain, buildings) produces a negative value or "error feeling".

This principle is used for collision avoidance with any physical or virtual obstacle. Physical obstacles may be other aircraft, mountains, the ground, buildings, aerials and so on. Virtual obstacle means a prohibited or unadvisable area like national airspace, military zones or areas with hazardous meteorological conditions. Therefore, a number of different exercises were run with real and virtual obstacles. The aircraft moves around the obstacle/terrain and steers again towards the previous target.

Both when avoiding flight into terrain or contouring physical and virtual obstacles the agent showed great flexibility. The terrain avoidance or contouring maneuvers were not only "up or down" actions but mixed with left and right turns at many different and versatile angles, adapting the maneuver to the situation.

Additionally, this software can select either a contouring maneuver, tracing the obstacle's profile, when it is close to the terrain or a virtual obstacle, or another longer and straighter trajectory to avoid an obstacle that is at a distance. This flexibility is very important for smart and safe aircraft behavior.

A third kind of experiment was run with two aircraft. The software receives negative values when another aircraft approaches the agent. The two aircraft approaching with opposite trajectories avoid each other in a smart and flexible way. The algorithm is so effective, from a safety point of view, that there is never a collision even in the very early trials in the simulation. This is not done automatically today with the help of TCAS II or T2CAS, since the aural and visual warnings alert the pilot who has to do the maneuver manually [4]. Additionally, the software maneuver is very flexible; it is not only an "up or down" motion but is adapted to the situation. The vertical action is by far the fastest, but there may be another aircraft coming behind and above the other, which makes such a maneuver unadvisable. This kind of situations may happen frequently with a very short reaction time with military aircraft.

In addition to the collision avoidance maneuver at a closer aircraft distance, the software provides a conflict avoidance action, where, when possible, a straighter trajectory is decided at a longer distance to avoid a future conflict with another aircraft [10].

To make it more difficult, two parallel aircraft were positioned near the airport to land on the same runway. This constitutes a permanent risk of collision. At the beginning of the learning simulation phase, there are of course negative values (because they move too close to each other), as well as a lack of equilibrium and stability. The learning process leads to the ideal behavior, where both aircraft keep together, proceed in parallel, optimize the two trajectories and glideslopes and, finally, give each other priority to land. Since the software never gives any landing priority to any aircraft, they have to reach agreement with each other and try to optimize their behavior depending on their trajectories and positions.

To help with this and other experiments with several aircraft, the process has been designed so that what one agent/aircraft learns is transmitted to the other for faster learning and better performance. In this experiment, the two aircraft approach the runway along two parallel trajectories, at the minimum permitted distance. They proceed keeping this minimum distance and the due orientation until the aircraft furthest to the side is on the verge of moving out of the virtual cone. At this point the aircraft turns to correct its trajectory and is bypassed by the other aircraft, which goes straight on. Afterwards when the second aircraft is trying to avoid hitting the first one when it turns towards the center of the virtual cone, we find that the second aircraft slows down to avoid flying at less than the minimum distance. In this skilled way they give way to each other when necessary, respect the minimum distances, establish their own landing priorities and land one after the other at the start of the runway. This experiment has been successfully run with randomly selected aircraft starting points. The orientation of the aircraft was also randomly selected. Note that neither agent/aircraft knows the other's *intention*; even so, they never collide from very early on in the learning process (Figure 3). The learning curve converges at around 50,000 trials.

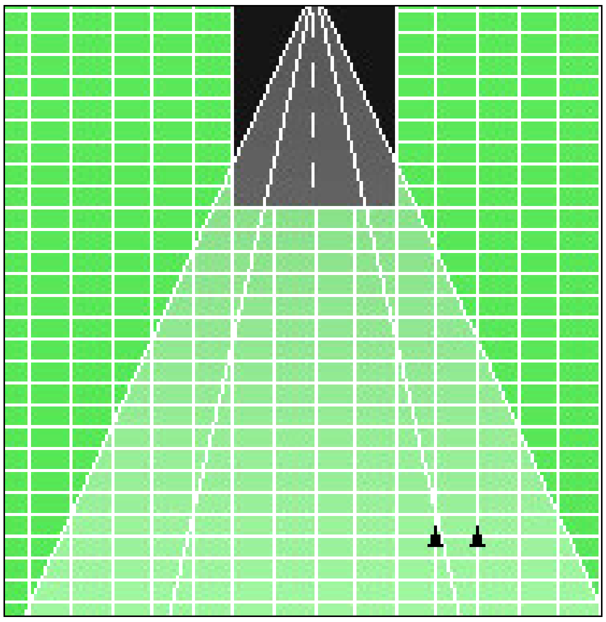
Fig. 3. Two small aircraft approaching the start of the runaway.

A very special obstacle experiment, which is difficult even for a human pilot, has been developed with a series of towers and buildings that the aircraft has to avoid. Several lines of towers have been positioned so that they can only be avoided by flying zigzag. Eventually the aircraft –an ultralight for the proportions shown in the figure- follows a zigzag flight path and goes around the building, avoiding collision with them all.

Later, an airport runway was added to the simulation environment. In this case, the aircraft has to go around the towers while descending towards the airport. The agent/aircraft moves successfully left and right around the towers and when it has successfully moved around the last tower, it steers again towards the start of the runway and finally lands (Figure 4).

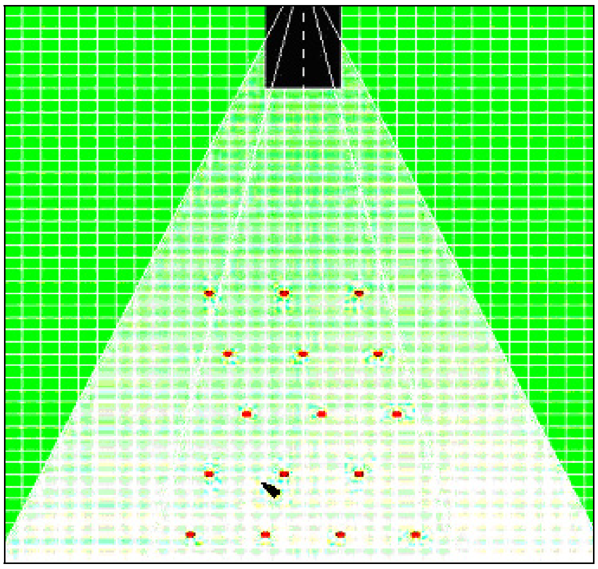
Fig. 4. Ultralight aircraft avoiding collision with towers.

The high flexibility of the maneuvers –always adapted by the FMS to the physical capabilities of each aircraft– is potentially useful in other scenarios where high density traffic, as well as a fast reaction time, is required.

This building collision avoidance capability can be helpful for designing an automatic procedure against aircraft hijackers. The pilot could quickly and easily key a short code or press just one special key, which would switch the aircraft to fully automatic mode without him being able to switch back (ground ATC could do this). In fully mode, it would be impossible to collide with a building or change the route.

Finally, the agent/aircraft is capable of navigating from one spatial point to any other. When a target is reached a new objective may be added and so on. This automatically provides a full navigation plan composed of a number of successive targets and a final objective.

4 Future Work

The exercises and tests that we have described here have been run on a simulator specially designed for this application. This software simulator logically accepts and receives digital data about the aircraft's environment and its action in this environment.

The final objective is to provide the algorithm with digital input data in the format in which it is actually relayed to the aircraft, instead of the simulated environmental data. Today most input information comes in digital format. EGPWS [5] supplies data on distance to the ground and TCAS II or T2CAS [4] provide distance to other aircraft, their trajectory and so on, sending both aural and visual signals to the pilot. Here the objective would be to supply the algorithm with digital input data – available before the signals. After the algorithm has processed the above data, it would trigger the automatic maneuver. The algorithm would benefit in this way from its generalization and be able to use I/O data in a different format.

One way of doing this, now under investigation, is with a flight simulator like the ones used today for pilot training and assessment. The software (except the I/O simulated data today) and the algorithm would take the pilot's place, actually dealing with the digital input data and triggering the output maneuvers. The results would be displayed on the screen as for a real pilot, and the learning process could be monitored.

Eventually, the fully automated navigation and collision avoidance system would be installed in the aircraft as TCAS II/T2CAS [4] and EGPWS [5] are today. We consider this to be a better place for the whole system for safety reasons, and mainly to achieve due reaction speed [2]. This does not preclude the additional use of environmental information, which could be provided from ground ATC for larger areas than are detected by the aircraft sensors.

5 Conclusion

Nowadays, there is a general trend towards automation in aeronautics. Automation is a possible way to protect safety in a growing air traffic scenario. Today, input information relayed to the aircraft is processed automatically instead of going through pilot assessment and judgment. Maneuvers are automatically recommended to the pilot, if not taken by the system. However, there are no automatic maneuvers in a number of cases. The pilot receives instructions on what to do but not exactly how he or she should do it.

This software, and its algorithm, has shown its capability to automatically produce intelligent and flexible maneuvers in the above cases using the simulated I/O data described in this paper. Since the algorithm is generalized for any I/O data or data in any format, we are planning to use it with flight simulator and aircraft I/O data.

This software provides for collision avoidance in mid-air with other aircraft, terrain or ground, buildings and so on. Mid-air collision avoidance is especially useful for military aircraft. This software could also avoid, or at least reduce human errors which are frequently due to misunderstandings between the cockpit and ground ATC.

This software provides very flexible and intelligent maneuvers adapted to the situation and not only "up or down" actions, which could be unadvisable in some cases. Additional flexibility is provided for moving around closer or not so close obstacles or terrain. Further capability is provided for either conflict prediction, triggering a straighter trajectory, or collision avoidance at a shorter distance.

The software incorporates additional capabilities for contouring national airspace, military zones or areas with dangerous meteorological conditions. The software may allow a faster reaction time for maneuvering in critical circumstances, discounting pilot reaction time. Finally, it offers a possible procedure against aircraft hijack by keying in a simple code, which switches the aircraft to fully automatic mode.

References:

[1] ACSS. L-3 Communications & Thales Company. T2CAS Terrain and Traffic Collision Avoidance System. ACSS, 2007. [Online]. Available: http://www.acssonboard.com/products/t2cas/.

[2] *ATC Magazine*. Satellite Navigation Systems. July 2006.

[3] *Aviation Week and Space Technology*. A350 Cockpit to introduce Flight Deck Enhancements. February 2006.

[4] Honeywell Aerospace, Avionics & Electronics, Products & Services. Honeywell, 2007. [Online]. Available: http://www.honeywell.com/sites/aero/Egpws-Home.htm.

[5] *MAAFAS Project* (More Autonomous Aircraft in the Future ATM System). February 2000.

[6] R. A. Paielli and H. Erzberger: Conflict Probability Estimation for Free Flight. NASA Ames Research Centre. *Journal of Guidance, Control and Dynamics*, vol. 40, no. 3, May-June 1997.

[7] R. S. Sutton and A. G. Barto: *Reinforcement Learning*. MIT Press, Cambridge, MA, 1998.

[8] R. S. Sutton, Generalization in Reinforcement Learning: successful examples using Sparse Coarse Coding. D. S. Touretzky, M. C. Mozer and M. E. Haselmo (Eds), *Advances in Neural Information Processing Systems*: Proceedings of the 1995 Conference, MIT Press, pp. 1038-1044. Cambridge MA, 1996.

[9] S. Álvarez de Toledo and J. M. Barreiro: Bioinspired Framework for General-Purpose Learning. *Lecture Notes in Computer Science* 1606: pp. 507-516. 1999.

[10] S. Álvarez de Toledo: *General Framework for Learning Work*. PhD Thesis, School of Computing, Technical University of Madrid. December 2000.