ARL-TR-20

AR-007-078

AUSTRALIA

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

AERONAUTICAL RESEARCH LABORATORY

MELBOURNE, VICTORIA

Technical Report 20

APPLICATION OF THE *A** ALGORITHM TO AIRCRAFT TRAJECTORY GENERATION

by

M.E. HALPERN

Approved for public release.

JULY 1993

AR-007-078

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
AERONAUTICAL RESEARCH LABORATORY

Technical Report 20

APPLICATION OF THE *A** ALGORITHM TO AIRCRAFT TRAJECTORY GENERATION

by

M.E. HALPERN

SUMMARY

Work carried out on the development of concepts for an automated trajectory generator for military aircraft is described. The system uses the A optimisation algorithm to design a ground track which represents an optimal tradeoff between a short path and a path over low terrain. This is achieved by minimising a cost function which is user selectable.*

This work incorporates path segments consisting of circular arcs, the design of an appropriate cost function and heuristics, as well as the inclusion of threats. Measures taken to improve the computational speed of the system are also described. Some examples of trajectories generated by the system are presented.

POSTAL ADDRESS: **Director, Aeronautical Research Laboratory**
506 Lorimer Street, Fishermens Bend
Victoria, 3207, Australia.

CONTENTS

1. INTRODUCTION

The automated design, in real time, of flight paths for low-flying combat aircraft is of interest since it allows the potential for improved mission effectiveness without increasing pilot workload. Automated route planning also allows the possibility of updated flight paths being generated in response to the receipt of new intelligence concerning the mission. Such flight paths could be presented on a display or used to drive an autopilot. Halpern [1] has surveyed recent work in the area of trajectory generation. As discussed in that work, an important approach for the trajectory design problem has been to represent the design problem as a state space search problem and to use an appropriate optimisation algorithm, commonly dynamic programming or *A** (pronounced A-star), to solve this. This approach has been used in several projects (see [1]). This approach requires reduction in the size of the search space in order to make the problem computationally tractable for reasonable flight distances.

To demonstrate the use of a state space search algorithm for terrain avoidance (TA) and to provide a starting point for future development, Selvestrel and Goss in Air Operations Division at ARL developed a PC (IBM style personal computer) based visualisation tool [2] which uses the *A** algorithm to generate optimal ground tracks over a patch of terrain represented by a grid of terrain altitudes from a terrain database. The software was written in the C programming language.

A ground track produced by the system consisted of connected straight segments, each oriented at one of sixteen headings. The path minimised a cost function which was a weighted sum of two components. One component was the total path length and the other was the sum of the path segment lengths, each weighted by the terrain altitude below the segment. Minimisation of this cost function encouraged the resultant path to be a trade off between a short one and a longer one over lower terrain.

This Technical Report describes the development of that tool into a form closer to a deployable system and is organised as follows. Section 2 contains some background on the *A** algorithm and a description of the modifications that were necessary for this work. Section 3 describes the system and the development work carried out, including the use of path segments made of circular arcs, the selection of cost functions and heuristics used to drive the algorithm, and the implementation of specific threats. Section 4 describes the methods used to reduce the search space to allow larger problems to be solved in a reasonably short time. Section 5 contains some sample trajectories and some results indicating the effectiveness of some of the techniques presented in Section 4.

2. THE *A** ALGORITHM

2.1 Standard Algorithm

The standard algorithm is outlined here and presented in algorithmic form in Appendix 1. Readers are referred to Pearl's book [3].

In its standard form, the algorithm finds the optimal allowable path from a starting node, S, to a goal, G through a subset of a set of fixed nodes. The goal may consist of more than one goal node, in which case any of these may be the endpoint of the optimal path. The path is optimal in that it minimises a cost function supplied by the designer (so that when this cost function is evaluated over any other allowable path from start to goal, its value will not be smaller). Here, allowable means that the path formed by the interconnection of path segments does not violate constraints supplied by the designer for the particular problem. In this work, the term optimal is used to imply optimal allowable. A conceptually simple approach to solving such an optimisation problem would be to generate all allowable paths and then to test them to find which is optimal. The sequence of operation of the *A** algorithm involves generating nodes which may turn out to be on the optimal path and testing them to see how they compare with the others.

The *A** algorithm proceeds by building up paths from starting node S. The nodes on these paths are stored on two lists, called 'OPEN' and 'CLOSED'. The endpoint of each path is stored in OPEN. The other nodes on each path are stored in CLOSED. Each node has a pointer to its predecessor on the path. The pointers point back rather than forwards because a path may branch. Each node, n, also has a value of a node evaluation function $f(n)$ associated with it.

This node evaluation function is an estimate of the optimal cost of a path from S to G through node n. This evaluation function is the sum of two parts: $f(n) = g(n) + h(n)$. The first part, $g(n)$, is the current lowest cost of a path from the start to node n. The second part, $h(n)$, is an estimate of the optimal cost from node n to the goal node G. If this estimate, called a heuristic, is always (that is, for every node n) an underestimate of the true optimal cost from node n to the goal then the heuristic is said to be admissible, and the algorithm will find the lowest cost path joining start and goal. An admissible heuristic which is larger (is a closer underestimate of the true optimal cost from node n to goal) than another heuristic is said to be better informed than the other one.

At each step of the algorithm, the node n on OPEN, with lowest value of $f(n)$ is expanded (ie has some single new path segments ending at nodes, called its children, emanate from it) so that its children will become path endpoints to be placed on OPEN where their values of f will be subsequently compared with the others. Node n is then placed on CLOSED since it is no longer a path endpoint.

The algorithm propagates in this way, using the node evaluation function to select the node

from OPEN to be next expanded. If, at some time, a node generated coincides with a node already on OPEN or on CLOSED, their node evaluation functions are compared and only the child with the lower cost is retained. This is done by giving it a pointer to the lowest cost path back to start. If a node on CLOSED has been updated, it is removed from CLOSED and placed on OPEN so that is is again an endpoint. Nodes which point back to this node will then be automatically updated as the algorithm proceeds.

Eventually, there will be a goal node on CLOSED. This signifies that the lowest cost node on OPEN was at the goal and the algorithm terminates. The optimal path is then found by backpropagation, that is by following the pointers back from the node at goal all the way to start.

2.2 Inadmissible Heuristics

A well informed heuristic is desirable because it involves fewer node expansions than a less well informed heuristic. However, a better informed heuristic is usually more difficult to calculate than a less well informed one. There is thus a trade-off between time spent calculating a good heuristic which will result in fewer node expansions and time spent expanding more nodes and thus manipulating longer lists of nodes if using a simpler less well informed heuristic.

In many applications, one is prepared to sacrifice a guarantee of optimality for improved speed of solution. This may be achieved by using an inadmissible heuristic, that is, one whose values are not always an underestimate of the optimal cost from node n to the goal. Approaches which use an inadmissible heuristic but guarantee to find a solution with a cost within a specified tolerance of the optimal solution are given in [3].

The use of a node evaluation function which incorporates information about what is going on both fore and aft of node n is the principal difference between the A^* algorithm and dynamic programming, which uses information from only one side. For example, backward dynamic programming calculates costs from node n to goal and uses no information about what can occur between start and n to guide the search.

2.3 Algorithmic Modifications

As a consequence of the implementation of path segments consisting of circular arcs described in Section 3.1, it was found necessary in this work to modify the standard algorithm as follows. Each child generated was tested for coincidence only with a node on OPEN and not for coincidence with a node on CLOSED as in the standard algorithm. If coincidence with a node on OPEN was found, only the child with the lower cost was retained. The reasons for this modification are given in more detail in Section 3.1.

3. SYSTEM DESCRIPTION

The system was based on that in [2], but was ported to a Commodore AMIGA 3000UX computer running the AMIGADOS operating system. The terrain map was represented as a 250 x 250 array of terrain heights, each associated with the altitude of a square area of terrain called in this work a grid square. The grid square sizes were made variable and the length of a side was denoted *grid_size* in metres (m). The two sizes mostly used were 100 m and 480 m. Using a grid square of side 100 m gave a terrain patch of 25 km x 25 km. This was the size used for most of the work described here. A node is defined by its position in the terrain patch and by the heading of the path segment at that position. The locations of the start and goal nodes were specified by the user and as the algorithm ran, those nodes which were being expanded were displayed. This visualisation capability was useful for developing methods for improving performance since it gave a dynamic indication of the progress of the algorithm.

3.1 Circular Path Segments

The optimal path consists of segments each of which begins and ends at a node. In [2], the centre of each grid square defined the (x,y) coordinate of a possible node. Each path segment thus began and ended at the centre of a grid square. The path segments were straight lines of fixed length at one of sixteen orientations. Each path segment was able to branch into three path segments at a node. The headings of the new segments were selected according to the heading of the previous segment. The three out of the sixteen possible directions chosen were straight ahead, and the gentlest left and right turn (approximately 22.5 degrees). These moves were designed to correspond with path segments having one g lateral acceleration for three seconds of level flight at a speed of 200 metres per second (m/s).

In the work reported here, these straight line segments were replaced by circular arcs with any orientation allowed and any starting and end points, relative to the grid squares. The length, l (m), of the segment, the velocity, v (m/s), and lateral acceleration in units of gravitational acceleration were specified by the user. The lateral acceleration is converted to m/s^2 and denoted a_{lat}. The geometry of a left turn segment is as shown in Figure 1.

Note from Figure 1 that γ, the change in heading, is also the angle subtended by the arc at its centre. We have

$$a_{lat} = v^2/r$$

so

$$r = v^2/a_{lat}.$$

Also,

$$\gamma = l/r$$
$$= l\, a_{lat}/v^2.$$

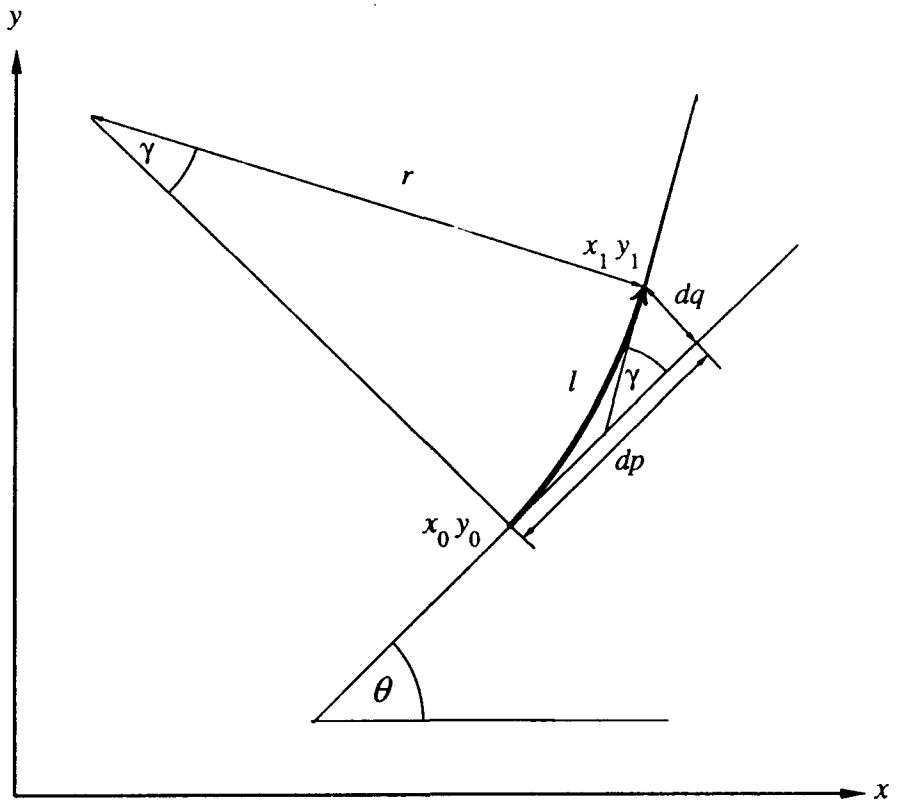

Figure 1. Geometry of a left turn segment.

Displacements, dp (m), in the direction of the incoming heading, and dq (m), orthogonal to it, may then be found. We have

$$dp = r\sin\gamma$$

and

$$dq = r(1-\cos\gamma).$$

Note that dp and dq need only be calculated once for each type of segment.
The coordinates at the end of the segment are then given by

$$x_1 = x_0 + dp\cos\theta - dq\sin\theta$$

and

$$y_1 = y_0 + dp\sin\theta + dq\cos\theta.$$

For a right turn segment,

$$x_1 = x_0 + dp\cos\theta + dq\sin\theta$$

and

$$y_1 = y_0 + dp\sin\theta - dq\cos\theta.$$

Node positions and headings calculated using these formulae are continuous variables and the likelihood of two partial paths ending at the same point is very small. This means that coincidence with nodes already calculated will rarely occur. In order to reduce the number of nodes used in the search, node positions were quantised to the containing grid square and headings were quantised to one of sixteen equal arcs of a circle. These quantised

values were then used for testing for coincidence with a node on OPEN as described in Section 2. Note that the real values of position and heading and not the quantised values were used to calculate the path segments. This enables a smooth path to be calculated while discarding some unnecessary nodes.

The standard A^* algorithm as described in Section 2.1 involves testing nodes for coincidence on CLOSED as well as on OPEN. Testing for coincidence with a node on CLOSED is inappropriate for this work since it is possible for a node n on CLOSED to have successors which would not be updated if node n were replaced by a node with identical quantised heading and position. This is because two nodes with identical quantised heading and position could be sufficiently different that their successors are not equivalent when quantised. If this occurred, the successors of a node replaced on CLOSED would still point back to the old node so that the program could crash. The problem does not occur with nodes on OPEN since they are path endpoints. A consequence of the lack of checking for coincidence on CLOSED is an increased number of nodes on CLOSED, but our experience has been that this does not greatly slow down the algorithm.

The grouping of similar (heading, position) points and considering them as equivalent must mean that some of the results which apply for a 'pure' A^* implementation are weakened for this work. Nonetheless, the approximation seems to be a reasonable one and, from examinations of the trajectories produced, the approach seems to work well.

Moves were specified as having one out of five possible directions. From left to right, these were *hard left*, *soft left*, *straight*, *soft right* and *hard right*. At each node, the three possible moves were the same again and one to the left and right. For example, if the incoming move to a node was soft left, then the three possible outgoing moves were soft left, hard left and straight.

3.2 Cost Function

The trajectory design has been formulated in an optimisation framework and the cost function is selected by the designer to obtain the type of solution required. This approach is well established in engineering design. For example, in many optimal control system design approaches, the designer uses selections of cost function to 'drive' the design. Such a framework is typically used to allow the designer to trade off various design criteria in a systematic manner.

In this work, we are interested in minimising risk to the mission from exposure to threats. It is known, e.g. [4], that in the absence of specific threat information, low flight over low ground is a sensible policy. In [4], in the generation of a three dimensional path, Asseo has penalised the square of the aircraft altitude above sea-level in order to to minimise the square of the expected kill probability in a region inhabited by uniformly distributed surface to air missile sites whose exact locations are unknown. This indicates a need for

the cost function to contain a term which penalises flight over high ground.

However, to avoid excessively long paths, it may be necessary to penalise path length. A way of incorporating these often conflicting requirements into the path design is to choose the cost function to be a linear combination of path length and path length weighted by terrain altitude below the path. This encourages the path to be short and over low terrain. The cost function, J, to be minimised, may be written as

$$J = J_d + J_e + J_t \tag{1}$$

Here, J_d penalises path length and is of the form

$$J_d = k_d \sum_i l_i \tag{2}$$

where k_d is a positive weighting chosen to select the importance of path length and l_i is the length of the ith path segment. The summation is over the path from start to goal.

Also, J_e penalises path sections over high altitude terrain and has the form

$$J_e = \frac{k_e}{75} \sum_i l_i (e_i - e_{ref}) \tag{3}$$

where

- k_e is a positive weighting chosen by the designer to reflect the importance of a path over low ground (the division by 75 allows convenient values for k_e for the terrain used in this work),
- e_i is the average terrain altitude under the i'th path segment and is obtained by sampling terrain altitudes at a specified number of points under the segment, and
- e_{ref} is a reference altitude whose function is to reduce the effect of path length on J_e.

The selection of e_{ref} is problematic. The value used for e_{ref} was the altitude of the lowest terrain grid square in the search area.

The threat component, J_t, of the cost, penalises path sections near specified threat locations and is described in Section 4.4.

Equations (2) and (3) have the feature that if the segment lengths or the number of points under a segment used to calculate e_i are changed, it is not necessary to readjust the values of k_d and k_e to maintain the trade-off between path length and path over low terrain.

3.3 Heuristics

If the standard A^* algorithm is used and if the heuristic is admissible, then the form of the heuristic will affect the speed at which the algorithm runs to completion but will not affect the final result. This guarantee of optimality with an admissible heuristic is not of great importance for many practical problems since a solution path close to the optimal one will often be adequate. Nonetheless, initially we examine the design of some admissible heuristics.

It is natural to decompose the heuristic into components which correspond with those of the cost function, J. That is, we decompose $h(n)$ as

$$h(n) = h_d(n) + h_e(n),$$

where $h_d(n)$ is the heuristic for the distance component and $h_e(n)$ is the heuristic for the elevation component. A heuristic $h_t(n)$ for the specified threats could be constructed along the same lines as $h_e(n)$ since threats were modelled as areas of high terrain (see Section 4.4).

For J_d, a simple heuristic is

$$h_d(n) = \mu_d k_d l_{nG}$$

where

μ_d is a positive scaling factor with value less than or equal to 1.0 for an admissible heuristic or a larger value for an inadmissible one,
k_d is as used to calculate J_d in eqn(2), and
l_{nG} is the Euclidean distance from node n to the goal.

A more sophisticated heuristic might replace l_{nG} with the shortest possible path length from node to goal, taking into account the heading at the node, the required heading at goal, and any 'no-go' areas while respecting turn rate limits.

The design of a well informed h_e is more difficult. One may initially be tempted to consider the elevation cost of the shortest path from node to goal, but a heuristic based on such a path may not be admissible if the optimal path is longer and over lower terrain.

One method for obtaining an admissible elevation cost heuristic involves dividing the map into regions, r_i, where i = 1, 2, ... , as illustrated in Figure 2, and summing an underestimate of the elevation cost to traverse each region. With this approach, the regions must be chosen in such a way that they are all traversed at least once by any path from the start to goal. The regions will then also have the property that given the position of any node n, a set of regions which must be crossed by any path from node n to the goal may be determined. Suitable regions could be bounded by concentric circles or squares surrounding the goal or the start. It is better to choose a set of regions surrounding the

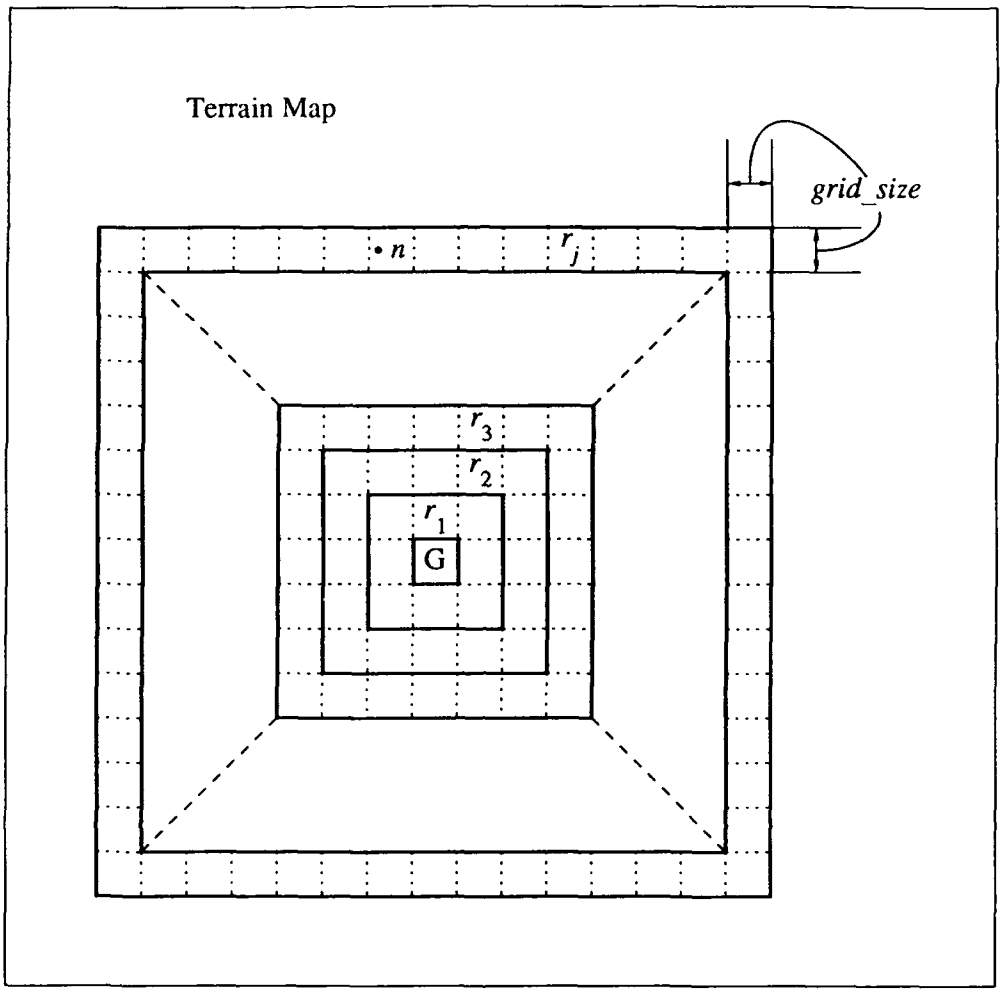

Figure 2. Illustration of terrain map decomposition into regions for calculation of elevation heuristic, $h_e(n)$.

goal because we wish to calculate heuristics for paths to the goal. The regions are numbered so that region r_{i+1} surrounds region r_i with r_1 being the smallest region surrounding the goal.

If the minimum elevation cost to traverse region r_i is called α_i, then an admissible (provided positive scale factor $\mu_e<1.0$) elevation heuristic for every node n in region r_j is given by

$$h_e(n:n\in r_j) = \mu_e \sum_{i=1}^{j-1} \alpha_i$$

The use of regions bounded by concentric squares based on the terrain grid squares is illustrated in Figure 2. This arrangement allowed the α_is to be calculated simply.

With the map decomposition shown in Figure 2, α_i is given by, from eqn (3)

$$\alpha_i = \frac{k_e}{75} \; grid\_size \; (e_{min}(j) - e_{ref})$$

where $e_{min}(j)$ is the minimum terrain altitude in region r_j, and k_e and e_{ref} are as in eqn (3). At large distances from the goal, the regions are not only large but consist of grid squares which are far from each other so their (the regions') contribution to the heuristic is likely to be a low underestimate since the value of $e_{min}(j)$ could depend on a low altitude grid square far from the optimal path. To reduce this effect, an alternative arrangement involving squares emanating from each node, n, as well as from the goal was considered. The idea here was that the heuristic would be better informed if based on regions bounded by smaller squares than shown in Figure 2. This scheme was implemented but was discarded since it involved a great deal more computation and gave very little reduction in the number of nodes expanded.

3.4 Specified Threats

Specified threats were treated as areas of high ground with their own penalty weighting distinct from that used to penalise path length over high ground. Regions under the influence of specified threats were represented by two concentric circles as shown in Figure 3. The equivalent terrain altitudes were specified by the user in the two regions, one inside the inner circle (the kill region) and the other between the two circles (acquisition region). This allowed the user to apply a more severe penalty to a path over the inner region of the threat.

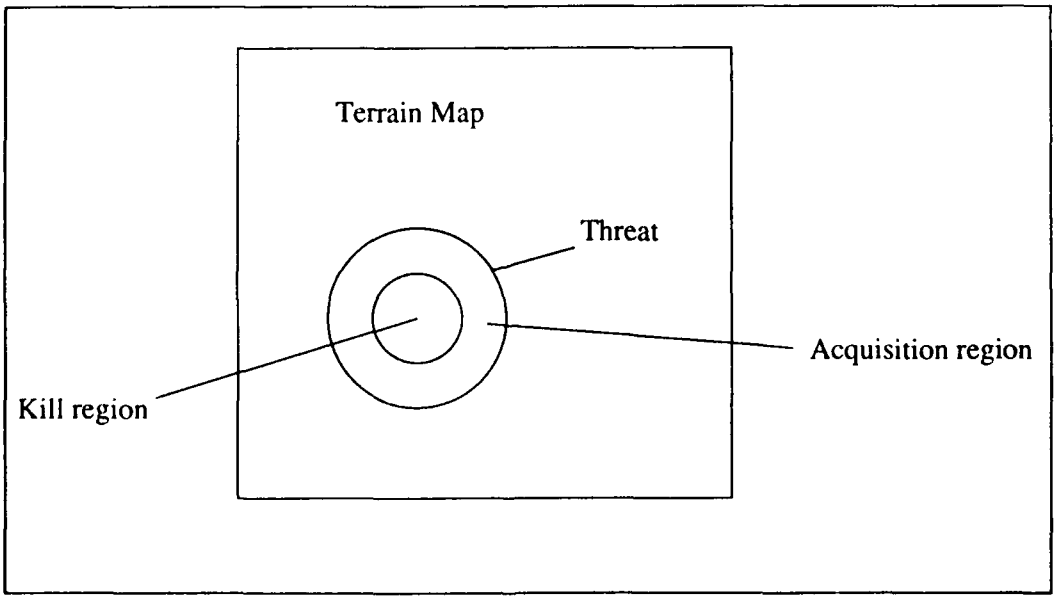

Figure 3. Plan view of area of influence of a specified threat.

4. TECHNIQUES USED FOR REDUCING SEARCH SPACE

As indicated previously, it is necessary to prune the search space in order to solve large problems in a reasonably short time. The visualisation tool [2] proved useful in highlighting situations where the algorithm was spending time unproductively. As indicated in Section 3, the A^* algorithm proceeds by generating nodes which may turn out to be on the optimal path and then testing them. Reducing the search space involves reducing the number of nodes generated.

4.1 Use of an Inadmissible Heuristic

Pearl [3] indicates that in many problems, the A^* algorithm spends much time discriminating between solutions which appear to be almost equally good. This appears to be exacerbated if a poorly informed heuristic is used. Making the heuristic larger leads to a solution being obtained more quickly, but if the heuristic is not admissible, the guarantee of optimality is lost. In this work, the elevation heuristic, h_e, incorporates a scaling factor, μ_e, and the distance heuristic has a scaling factor μ_d. These each have nominal value 1.0 and can be increased to obtain an inadmissible heuristic. The performance improvement available is discussed in Section 5.

4.2 Use of Bounded Number of Headings at Each Grid Square

A method for restricting the number of nodes generated is to discard nodes according to the number of quantised headings present in a grid square. Limiting the number of allowable quantised headings at each grid square to the first one or two chosen by the algorithm gave considerable improvement in run times with usually no degradation of path since the heading at a point on the optimal path often resulted from the first child there.

4.3 Use of Boxed Search Space

An obvious method for pruning the search space is to specify a region outside which the algorithm may not explore. A simple region is a rectangular corridor surrounding the straight line path of length, l, from start to goal. A guide to the width of the corridor was obtained by considering the relationship between the width, w, of the corridor and the length, $l(1+\Delta)$, of a simple path which touched the edge of the corridor midway between start and goal as shown in Figure 4.

From Figure 4, using Pythagoras, one readily obtains $w = l\,[\Delta\,(2 + \Delta)]^{1/2}$.

This was used in conjunction with a bound on the allowable heading deviation from a straight line joining start and goal.

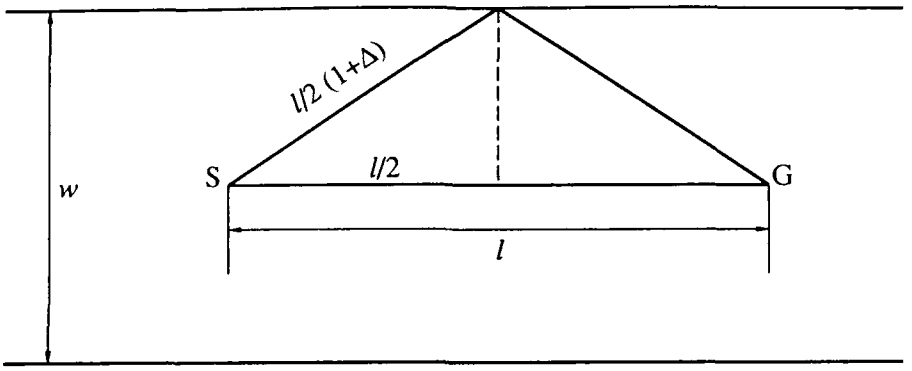

Figure 4. Illustration of search space reducing corridor based on a simple path of length $l(1+\Delta)$.

The performance improvement obtained by this technique is discussed in Section 5.

4.4 Use of Different Length Moves

The use of long moves in non-critical areas, that is away from the start, goal and threats, allowed improvements in run time without significant degradation of the solution. It was found necessary to make the long moves have large turn radii in order to avoid making large changes in direction. This process resulted in reduced manoeuvrability where the moves were long. The short moves were made to correspond with larger lateral accelerations.

The use of different length moves allows the possibility of randomly inserted short moves. This may be useful for designing routes less predictable by an enemy.

5. RESULTS

5.1 Sample Trajectories

This Section contains three sample trajectories, shown in Figures 5, 6 and 7. The same 25 km × 25 km patch of terrain was used for each of these. Some of the parameters used to generate all three paths are listed below.

(1) The three paths are each designed for a velocity of 200 m/s.
(2) The longer path segments have a length of 1200 m with hard left and hard right lateral accelerations of 2.0 g and soft left and soft right accelerations of 1.0 g.
(3) The shorter segments have a length of 600 m and hard and soft accelerations of 3.0 g and 1.7 g respectively.

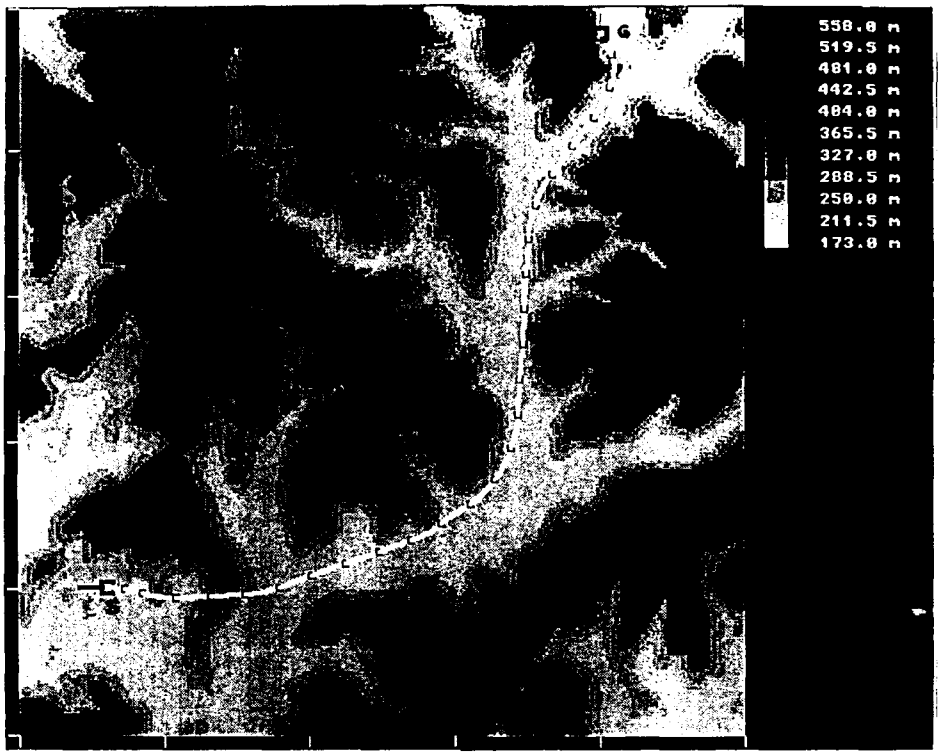

Figure 5. A trajectory with k_e=k_d=1.0. Terrain altitude (metres above sea level) indicated upper right.

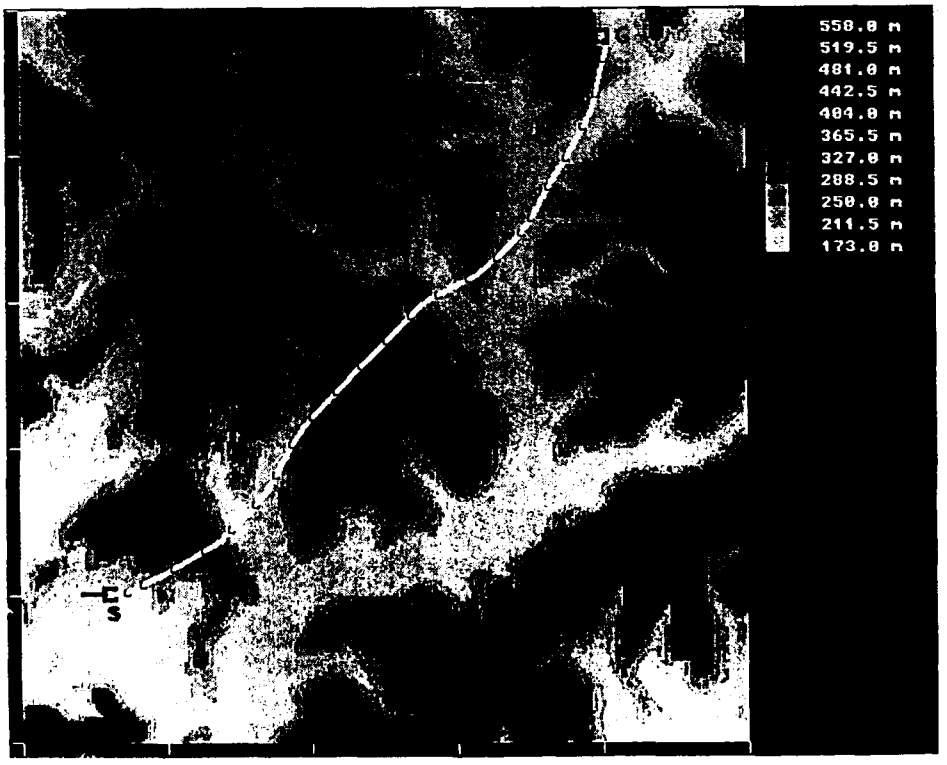

Figure 6. A trajectory with k_e=1.0 and k_d=10.0.

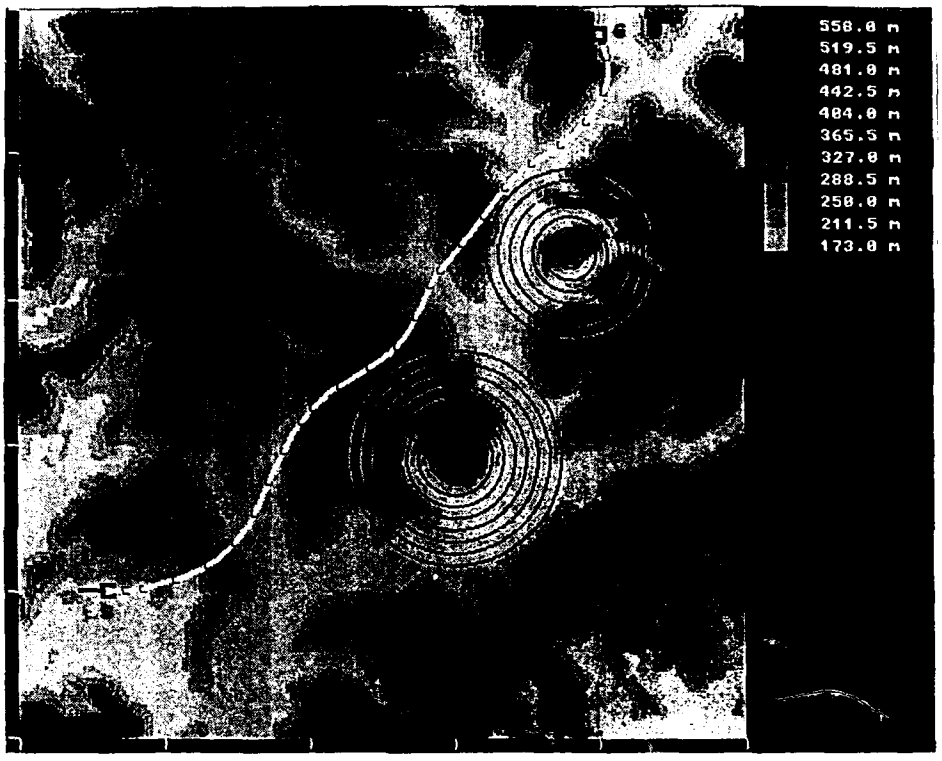

Figure 7. A trajectory with two threats modelled as 'no-go' areas (k_e=k_d=1.0).

In Figure 5, the trade-off between total path length and path over low terrain was achieved using $k_d = k_e = 1.0$. For this problem this results in a path which follows low terrain.

Figure 6 shows the trajectory produced with $k_d = 10.0$. This larger value of k_d increases the penalty associated with total path length and so causes the path to be shorter than in Figure 5.

Figure 7 shows the trajectory produced using $k_d = k_e = 1.0$ for a scenario with two threats modelled as 'no go' areas.

For all the runs described in this work, the number of nodes with different quantised headings in a grid square was limited to one, unless stated otherwise.

The trajectories generated have a length of order 20 km, which, at a speed of 200 m/s, represents a flight time of 100 seconds. Calculation times considerably less than this are achievable with the use of slightly inadmissible heuristics.

If longer trajectories are required, waypoints may be introduced to subdivide the problem and the path may be designed by separately calculating each section between adjacent waypoints. In this way, the required computing time increases linearly with total trajectory length, rather than exponentially as would be the case if the large problem were solved in one section.

At this time, the price of computing hardware is still falling and faster computers are becoming more readily available. This helps to indicate that the system described in this work has potential real time application.

5.2 Use of an Inadmissible Heuristic

Table 1 shows the effect of using an inadmissible heuristic on the performance of a run like that in Figure 5. The Table shows clearly how the total number of nodes falls when the heuristic is made inadmissible through the use of μ_d>1.0 or μ_e>1.0. The cost tends to increase when these scaling factors are increased. In a pure A^* framework, one would not find a case where an inadmissible heuristic would give a lower cost than an admissible one. This type of behaviour can, however, occur in the work described here due to various deviations, described in this document, of the implementation from standard A^*. The tabulated run times do not include the time needed to calculate the value of the heuristic for each grid square. This took approximately 8.0 seconds for the 250 x 250 grid square maps used.

Table 2 shows the effect of scaling up the heuristic for a problem like that in Figure 6.

μ_d	μ_e	total nodes	J	run time (s)
1.0	1.0	26839	42676.7	325.0
1.0	1.5	16182	42595.3	131.9
1.0	2.0	5843	43062.0	26.9
1.0	2.5	4947	43396.7	21.2
1.05	1.5	10069	42595.3	57.5
1.1	1.5	4527	42276.7	18.8

Table 1. Effect of inadmissible heuristic on the performance of a run like that in Figure 5.

μ_d	μ_e	total nodes	J	run time (s)
1.0	1.0	12096	293528.7	85.2
1.0	1.5	11681	294402.7	79.6
1.0	2.0	10503	294540.0	67.8
1.0	2.5	8876	294540.0	52.4
1.05	1.5	6919	294786.0	36.0
1.1	1.5	3490	294948.0	14.2

Table 2. Effect of inadmissible heuristic on the performance of a run like that in Figure 6.

Here, the effect of increasing μ_e is less than in Table 1 since the elevation component is a smaller part of the cost.

The differences in trajectory were small for small changes in cost and a reasonable choice of μ_e and μ_d for both of these runs is μ_e=1.5 and μ_d=1.05.

5.3 Use of Bounded Number of Headings at Each Grid Square

Table 3 shows the effect of increasing the allowed number of headings at each grid square for a run like that in Figure 5 with μ_e=1.5 and μ_d=1.05. The total number of nodes and run time are considerably worse for only a small reduction in cost with a bound of three headings per grid square.

max headings per grid square	total nodes	J	run time (s)
1	10069	42595.3	57.5
2	19378	42756.0	176.8
3	27375	42346.0	325.9

Table 3. Effect of bounding number of headings at each grid square.

Δ	heading deviation bound (deg)	total nodes	J	run time (s)
10.1	180	16182	42595.3	131.9
10.1	60	10883	42644.0	67.3
0.2	180	14828	42595.3	111.2
0.2	60	11231	42644.0	70.2
0.1	180	17927	52596.0	122.4
0.1	60	14483	53531.3	89.7

Table 4. Effect of corridored search space and bounded heading deviation on the performance of a run like that in Figure 5.

5.4 Use of Boxed Search Space

The effect of a boxed search space and bounded heading deviation on a run like that in Figure 5 is shown in Table 4. For this problem, the approach is not very effective in improving performance when compared with scaling up the heuristic. When a small

bounding corridor was used, the algorithm generated many nodes with with large heading deviations. This tendency was reduced by bounding the heading deviation. Nonetheless, it seems a good idea to incorporate both kinds of bounds on the search space in a fairly unrestrictive manner.

6. CONCLUSION

This Technical Report describes work carried out on the development of concepts for an automated optimal trajectory generator based on the A^* algorithm, which is a search based algorithm for finding an optimal path through a state space. Here the algorithm is applied to the generation of ground tracks for military aircraft in a terrain avoidance role.

The general aim of the development was to bring the concept demonstration work reported in [2] to a form closer to a deployable system. This involved enhancing the flexibility and reducing the execution times of the algorithms to enable the design of long trajectories in a reasonable time. In particular, the development work included the incorporation of path segments consisting of circular arcs and the design of appropriate cost functions and heuristics. Other measures taken to improve the speed of the system were also described and evaluated. Some examples of ground tracks generated by the system are presented.

Worthwhile improvements in the time to design paths were obtained by scaling up the heuristic, which guides the search. Only minor deterioration of the path resulted. Methods involving the use of a corridor to bound the search space were found to be less effective in improving the performance of the path generator.

Consideration of the computational resources used in this work and methods for designing longer trajectories suggests the potential real time application of a system like the one developed here.

ACKNOWLEDGEMENT

The author would like to thank Robin Miller, Mario Selvestrel and Simon Goss for their valuable contributions to the work presented here.

REFERENCES

1. Halpern, M.E., 'Optimal Trajectories for Aircraft Terrain Following and Terrain Avoidance - A Literature Review Update', *ARL-TR-5*, DSTO Aero Res Lab, Melbourne, 1993.

2. Selvestrel, M. and Goss, S., 'A Visualisation Tool for State Space Approaches to Aircraft Flight Paths', ARL document in preparation.

3. Pearl, J., *Heuristics: intelligent search strategies for computer problem solving*, Addison-Wesley, Reading, MA, USA, 1984.

4. Asseo, S.J., 'Terrain Following/Terrain Avoidance Path Optimization Using the Method of Steepest Descent', *Proceedings of the 1988 National Aerospace Electronics Conference*, Dayton, OH, USA, pp. 1128-1136.

APPENDIX

The *A** Algorithm ([3], page 64)

1. Put the start node S on OPEN.

2. If OPEN is empty, exit with failure.

3. Remove from OPEN and place on CLOSED a node n for which f is minimum.

4. If n is a goal node, exit successfully with the solution obtained by tracing back the pointers from n to S.

5. Otherwise expand n, generating all its successors, and attach to them pointers back to n. For every successor n' of n:

   a. If n' is not already on OPEN or CLOSED,
   estimate $h(n')$
   (an estimate of the cost of the best path from node n' to some goal node),
   and calculate $f(n')=g(n')+h(n')$
   where $g(n')=g(n)+c(n,n')$ and $g(S)=0$.

   b. If n' is already on OPEN or CLOSED, direct its pointers along the path yielding the lowest $g(n')$.

   c. If n' required pointer adjustment and was found on CLOSED, reopen it.

6. Go to step 2.

In step 5a, $c(n,n')$ is the cost to travel from node n to node n'.

For the work reported here, step 5 of the standard algorithm was replaced by the following.

5. Otherwise expand n, generating all its successors, and attach to them pointers back to n. For every successor n' of n:

   a. If n' is not already on OPEN,
   estimate $h(n')$
   (an estimate of the cost of the best path from node n' to some goal node),
   and calculate $f(n')=g(n')+h(n')$
   where $g(n')=g(n)+c(n,n')$ and $g(S)=0$.

   b. If n' is already on OPEN, direct its pointers along the path yielding the lowest $g(n')$.

   c. Do nothing here.

DISTRIBUTION

AUSTRALIA

Defence Organisation

Defence Central
- Chief Defence Scientist
- AS Science Corporate Management } shared copy
- FAS Science Policy
- Counsellor Defence Science, London (Doc Data sheet only)
- Counsellor Defence Science, Washington (Doc Data sheet only)
- Scientific Adviser Defence Central
- OIC TRS Defence Central Library
- Document Exchange Centre, DSTIC (8 copies)
- Defence Intelligence Organisation
- Librarian H Block, Victoria Barracks, Melb (Doc Data sheet only)

Aeronautical Research Laboratory
- Director
- Library
- Air Operations Division
  - Chief
  - Head Avionics Research
  - S. Goss
  - M. Selvestrel
  - Author: M. E. Halpern (6 copies)

Defence Science & Technology Organisation - Salisbury
- Library

Navy
- Navy Scientific Adviser (3 copies Doc Data sheet only)
- RAN Tactical School, Library
- CO AMAFTU

Army
- Scientific Adviser - Army (Doc Data sheet only)
- Director Army Aviation

Air Force
- Air Force Scientific Adviser
- Aircraft Research and Development Unit
  - Library
- OIC ATF, ATS, RAAFSTT, WAGGA (2 copies)
- CO SAN

Department of Transport & Communication
- Library

Other Organisations
- NASA (Canberra)
- AGPS

Statutory and State Authorities and Industry

ASTA Engineering, Document Control Office

Flinders
Library

LaTrobe
Library

Melbourne
Engineering Library

Monash
Hargrave Library

Newcastle
Library
Institute of Aviation

New England
Library

Sydney
Engineering Library

NSW
Physical Sciences Library
Library, Australian Defence Force Academy

Queensland
Library

Tasmania
Engineering Library

Western Australia
Library

RMIT
Library
Aerospace Engineering

University College of the Northern Territory
Library

SPARES (7 COPIES)

TOTAL (60 COPIES)

AL 149 DEPARTMENT OF DEFENCE

PAGE CLASSIFICATION
UNCLASSIFIED

PRIVACY MARKING

DOCUMENT CONTROL DATA

1a. AR NUMBER	1b. ESTABLISHMENT NUMBER	2. DOCUMENT DATE	3. TASK NUMBER
AR-007-078	ARL-TR-20	JULY 1993	DST 89/096

4. TITLE

APPLICATION OF THE *A** ALGORITHM TO AIRCRAFT TRAJECTORY GENERATION

5. SECURITY CLASSIFICATION (PLACE APPROPRIATE CLASSIFICATION IN BOX(S) IE. SECRET (S), CONF. (C) RESTRICTED (R), LIMITED (L), UNCLASSIFIED (U)).

DOCUMENT	TITLE	ABSTRACT
U	U	U

6. NO. PAGES: 21

7. NO. REFS.: 4

8. AUTHOR(S)

M.E. HALPERN

9. DOWNGRADING/DELIMITING INSTRUCTIONS

Not applicable.

10. CORPORATE AUTHOR AND ADDRESS

AERONAUTICAL RESEARCH LABORATORY

AIR OPERATIONS DIVISION

506 LORIMER STREET

FISHERMENS BEND VIC 3207

11. OFFICE/POSITION RESPONSIBLE FOR:

SPONSOR DSTO

SECURITY -

DOWNGRADING -

APPROVAL CAOD

12. SECONDARY DISTRIBUTION (OF THIS DOCUMENT)

Approved for public release.

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DSTIC, ADMINISTRATIVE SERVICES BRANCH, DEPARTMENT OF DEFENCE, ANZAC PARK WEST OFFICES, ACT 2601

13a. THIS DOCUMENT MAY BE ANNOUNCED IN CATALOGUES AND AWARENESS SERVICES AVAILABLE TO ....

No limitations.

13b. CITATION FOR OTHER PURPOSES (IE. CASUAL ANNOUNCEMENT) MAY BE [X] UNRESTRICTED OR [ ] AS FOR 13a.

14. DESCRIPTORS

Trajectory optimization
Algorithms
Military aircraft
Automatic flight control

15. DISCAT SUBJECT CATEGORIES

010401

16. ABSTRACT

Work carried out on the development of concepts for an automated trajectory generator for military aircraft is described. The system uses the A optimisation algorithm to design a ground track which represents an optimal tradeoff between a short path and a path over low terrain. This is achieved by minimising a cost function which is user selectable.*

PAGE CLASSIFICATION
UNCLASSIFIED
PRIVACY MARKING

THIS PAGE IS TO BE USED TO RECORD INFORMATION WHICH IS REQUIRED BY THE ESTABLISHMENT FOR ITS OWN USE BUT WHICH WILL NOT BE ADDED TO THE DISTIS DATA UNLESS SPECIFICALLY REQUESTED.

16. ABSTRACT (CONT).

This work incorporates path segments consisting of circular arcs, the design of an appropriate cost function and heuristics, as well as the inclusion of threats. Measures taken to improve the computational speed of the system are also described. Some examples of trajectories generated by the system are presented.

17. IMPRINT

AERONAUTICAL RESEARCH LABORATORY, MELBOURNE

18. DOCUMENT SERIES AND NUMBER	19. WA NUMBER	20. TYPE OF REPORT AND PERIOD COVERED
Technical Report 20	74 746F	

21. COMPUTER PROGRAMS USED

22. ESTABLISHMENT FILE REF.(S)

23. ADDITIONAL INFORMATION (AS REQUIRED)