Low Altitude Threat Evasive Trajectory Generation for Autonomous Aerial Vehicles

by

Ryan L. Pettit

B.S. Aeronautical and Astronautical Engineering
University of Washington, 2001

Submitted to the department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN
AERONAUTICS AND ASTRONAUTICS

at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June, 2004

Signature of Author: _____
Department of Aeronautics and Astronautics
June, 2004

Certified by: ___
Mark L. Homer
CSDL Technical Supervisor

Certified by: _____
Brent Appleby
Lecturer in Aeronautics and Astronautics
Thesis Supervisor and CSDL Technical Supervisor

Accepted by: _____
Edward M. Greitzer
H.N. Slater Professor of Aeronautics and Astronautics
Chairman, Committee on Graduate Students

Low Altitude Threat Evasive Trajectory Generation for Autonomous Aerial Vehicles

by

Ryan L. Pettit

Submitted to the Department of Aeronautics and Astronautics on May 13, 2004 in
Partial Fulfillment of the Requirements for the Degree of
Master of Science in Aeronautics and Astronautics

ABSTRACT

In recent years, high altitude unmanned aerial vehicles have been used to great success in combat operations, providing both reconnaissance as well as weapon launch platforms for time critical targets. Interest is now growing in extending autonomous vehicle operation to the low altitude regime. Because perfect threat knowledge can never be assumed in a dynamic environment, an algorithm capable of generating evasive trajectories in response to pop-up threats is required. Predetermination of contingency plans is precluded due to the enormity of possible scenarios; therefore, an on-line vehicle trajectory planner is desired in order to maximize vehicle survivability.

This thesis presents a genetic algorithm based threat evasive response trajectory planner capable of explicitly leveraging terrain masking in minimizing threat exposure. The ability of genetic algorithms to easily incorporate line-of-sight effects, the inherent ability to trade off solution quality for reduced solution time, and the lack of off-line computation make them well suited for this application. The algorithm presented generates trajectories in three dimensional space by commanding changes in velocity magnitude and orientation. A crossover process is introduced that links two parent trajectories while preserving their inertial qualities. Throughout the trajectory generation process vehicle maneuverability limits are imposed so that the resultant solutions remain dynamically feasible. The genetic algorithm derived provides solutions over a fixed time horizon, and is implemented in a receding horizon fashion, thereby allowing evasion of threat areas of arbitrary size. Simulation results are presented demonstrating the algorithm response for a rotorcraft encountering several different threat scenarios designed to evaluate the effectiveness of the algorithm at minimizing risk to the vehicle.

Thesis Supervisor: Brent Appleby
Title: Lecturer in Aeronautics and Astronautics

Acknowledgments

First off, I would like to thank my two advisors at Draper Laboratory, Brent Appleby and Mark Homer, without whom this work would never have been done. I would like to thank Brent for always being willing to take time out of his hectic schedule to meet with me, and for the many ideas that helped form and refine my research. I couldn't have picked a better research advisor. I am also extremely thankful to Brent and to Draper Lab for providing my research fellowship. If it hadn't have been for the financial support that I received from Draper I never would have been able to come to MIT. Likewise, I owe Mark for the many discussions we had about algorithms, for his constant stream of suggestions and support, for his great advice on preparing a thesis, and for keeping me on track. I can confidently say my research would not have been the same without his help.

And how could I forget my fellow DLF's and partners in suffering? Jason, Luke, Tiffany, Jillian, and Lucas – you guys made my time at MIT much more enjoyable than it would have been (although it never seemed like Lucas did any suffering...). Oh the countless hours spent with Jason, Luke, and Tiffany doing problem sets galore, many of which we never even turned in, but they built character, right? Thanks to Luke for opening up his house to his fellow DLF's on many occasions, and for reminding me of the bright side of things. I wish everyone the best of luck in what adventures lie ahead.

Thanks to Mom and Dad for putting up with all the unreturned phone calls and emails, and for reminding me to take a break every now and again. I would never be here if it weren't for your constant support in everything I undertake, and I thank you for all the sacrifices you made to make it possible for me to get here. Thanks to Dan and Suzie for teaching me the meaning of call screening, a skill I certainly put to use while at MIT, and for reminding me there's more to life than work.

And last, but not least, a special thanks to Tiffany for being there for me every step of the way. I never would have made it without you.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under Internal Company Sponsored Research Project C13130, Level 9 Autonomy, Contract IRD-04-2-6011.

Ryan L. Pettit Date

In memory of Arthur Garfield Pettit

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The past decade has seen tremendous progress in the capability of unmanned aerial vehicles (UAVs). High endurance UAVs of all sorts are available, and provide the ultimate in remote operation platforms. The lack of an in-vehicle human operator makes UAVs ideal for high risk missions, as well as those in which fatigue due to mission length preclude the use of a pilot.

Inevitably, UAVs have made their way into military battlefield operation, and have proven their value time and time again. The General Atomics Predator UAV has provided both a high altitude reconnaissance as well as remote weapons launch platform to great success. By outfitting the Predator with AGM-114 Hellfire electro-optically guided missiles, the vehicle has been able to not only gather reconnaissance from its cruise altitude of up to 25,000ft, but also strike time-critical targets without requiring deployment of forces [1]. The proven ability of the Predator in eight different military Operations since 1995 has placed the UAV in ever increasing demand, and has motivated interest in outfitting other UAV platforms with weapons systems [2, 3, 4]. Another such vehicle is the Northrop Grumman Global Hawk, which is capable of remaining aloft for over 35 hours at its cruise altitude of 65,000ft. The Global Hawk has likewise seen significant military use in recent years amassing over 1,200 combat flight hours in well over 50 missions [5, 6]. However, the very nature of the high altitude loitering

reconnaissance mission places the vehicle at risk of being shot down by surface to air missiles; indeed, such a mission turns the vehicle into the proverbial "sitting duck".

1.1 The future of combat UAVs

One of the best ways to mitigate this risk has been in use since the very dawn of combat aviation, and that is simply – fly low. An aircraft flying in close proximity to the ground is more difficult to detect, and even when detected, the response time available to the enemy can be severely restricted. Current fixed-wing UAVs mitigate risk of shorter range threats by flying at very high altitudes, however this strategy eliminates the potential for missions in which reconnaissance or weapons use require low altitude operation. In order to accommodate low altitude missions current rotorcraft tactics rely heavily on terrain flight, defined as "the tactic of using terrain, vegetation, and manmade objects to mask the aircraft from enemy visual, optical, electronic, and thermal detection systems" [7]. Based on this key notion, it is no surprise that current military thinking envisions the use of combat UAVs for low altitude penetration into enemy airspace. A team of UAVs could conceivably scout an area for unknown threats, provide reconnaissance, or strike key targets in order to make way for manned aircraft and ground personnel [8].

Indeed, these ideas paint a picture of the future of unmanned combat vehicles; however, such missions provide quite a challenge for the algorithms which autonomously operate the vehicles. Low altitude flight, and even more so Nap-Of-the-Earth (NOE) flight, is extremely demanding for a human pilot, typically requiring all of the pilot's facilities to avoid crashing. Likewise, an autonomous system's ability to operate at low altitudes is limited by its ability to generate flyable trajectories, and furthermore by the ability of the flight control system to track said trajectories. Of course, the vehicles could be remotely-piloted in order to reduce the complexity of the required autonomy algorithms; however, algorithms providing behavior such as threat response are required due to the possibility of loss of communication or to provide time for the human operator to assess the situation and review the threat response algorithm's plan.

In addition, these vehicles will be required to operate in partially unknown threat environments. Typically some known threat information will be available during the

initial planning of a mission; however, it is not good enough to construct vehicle trajectories off-line for the extent of the mission. It is common in mission planning to characterize a threat area by a region of high cost in order to detract the vehicle from entering the threat region. However, the initial ingress into hostile territory may be made *specifically* in order to provide intelligence data on hostile threats, in which case it is extremely unlikely that all threat locations or even types would be known *a priori* (indeed, such knowledge would invalidate the need for the mission). As new threat locations present themselves, any previously optimized plans run the risk of being invalidated due to the possibility of losing a vehicle to the new threat. This provides a significant hurdle to a potential planning system. It is computationally intractable to try and pre-plan a mission with contingencies for every different possible threat encounter scenario. Because of this, the planning system must be able to rapidly re-plan trajectories in order to avoid new threats encountered during the mission.

Beyond this, the possibility exists that the vehicle will become aware of a new threat while within the threat's range, in which case immediate action would be required in order to minimize the probability of losing the vehicle to enemy fire. The very essence of this problem requires potential response algorithms to operate in real-time. In such a circumstance the vehicle can respond in four general ways, being; (1) the vehicle continues along its initial path, with no update due to the impending threat, (2) the vehicle can choose to use its weapons to strike the new threat, (3) the vehicle can suspend its current mission plan and attempt to escape from the threat's range in a fashion that maximizes the potential of survival, or (4) the vehicle can remain diligent to its original plan, however it re-plans its path in an attempt to minimize the overall exposure to the threat while heading for its goal.

The first response option described above is clearly the least attractive response available, however if dynamic re-planning of the vehicle's trajectory is not available then it will be the only option. In the case of offline planning only, it is highly improbable that the trajectories determined before the knowledge of the impending threat would be the best means to ensure the probability of vehicle survival, and hence mission success. The second option is certainly viable, however it would have two certain implications. Firstly, if the vehicle is indeed carrying weapons that are intended for use in striking targets, use

of a weapon (or weapons) in order to destroy the new threat would jeopardize the ability of the vehicle to complete its mission. Indeed, a certain amount of re-planning would be required to determine whether the threat could be fired upon. Secondly, it is not currently the thinking of the military to allow an autonomous vehicle with weapons to decide to prosecute a target at will, with no human intervention. In order to reduce the risk of using weapons on ill-identified targets, a human operator would be required to assess the threat situation, and give the fire/no fire decision. This would inherently take a finite amount of time, and even were the turn and fight tactic to be chosen, some intermediate response would be required by the vehicle to reduce threat risk until the proper course of action can be taken.

This leads to the third and fourth response options. Both involve real-time generation of vehicle trajectories in order to minimize the probability of losing the vehicle to enemy fire, the only difference being in the latter case the original goal orientation is maintained. Such capabilities would certainly be highly desirable for a vehicle operating in an environment subject to pop-up threats, providing means for the vehicle to autonomously re-plan its trajectory in a timely fashion in order to evade the threat through maneuvering and terrain masking.

1.2 Objectives

The main contribution of this thesis is the development of an easily adaptable threat response algorithm for autonomous aerial vehicle trajectory generation in order to minimize the potential for vehicle attrition due to enemy fire. This is accomplished through trajectory refinement, exclusive of the use of countermeasures. The algorithm provides a vehicle trajectory which seeks to minimize risk to the vehicle through evasive maneuvering and explicit capitalization upon terrain masking in accordance to current combat tactics [7]. In so doing, the algorithm must account for dynamic maneuvering limits of the vehicle in order to produce flyable trajectories.

Furthermore, since the vehicle is assumed to be operating at low altitudes, an extreme importance is placed on providing trajectories that are fully four-dimensional (three spatial dimensions plus time) in order to maximize the vehicle's ability to reduce exposure to known threats. By exhibiting four-dimensional control over the vehicle the

algorithm can fully leverage both known information about the threat capability, as well as known terrain features which can serve to break line-of-sight (LOS), effectively preventing detection by the threat or breaking tracking if detected.

It is important to note the objective of this research is not to develop the full automation vehicle planning system, nor is it to develop the flight control system required to track a given trajectory. Rather, the intent is to identify and produce a threat response algorithm that is easily customized for many kinds of aircraft, and provides the required behavior based on a low altitude mission.

A genetic algorithm approach to solving the autonomous threat response trajectory generation problem is described in this thesis, and is analyzed for feasibility through the application to a representative low-altitude rotorcraft mission. The ability of the vehicle to seek out low risk areas is shown through scenario case studies of low altitude operation in the presence of threats while operating in mountainous terrain. This provides means to assess the advantages and disadvantages of solving the trajectory generation problem through the application of genetic algorithms. Note that genetic algorithms inherently offer many options for implementation, however the main objective of this thesis is to develop an appropriate algorithmic implementation and analyze its potential for solving the problem at hand, as opposed to optimization of the algorithm design itself.

1.3 Thesis outline

The organization of this thesis is as follows: Chapter 2 begins with an overview of the low-altitude mission, followed by a discussion of the specific threat evasion algorithm requirements. The chapter then goes on to review previous research in threat avoidance as well as trajectory planning, and discusses the motivation for the selection of genetic algorithms for threat response. Chapter 3 presents a general background into genetic algorithms, and continues with a specific description of the implementation used to address threat response re-planning. Chapter 4 presents a progressive set of case studies to demonstrate the effectiveness of the evasive response planner, and analyzes the quality of the resulting trajectories. Finally, Chapter 5 summarizes the conclusions made over the course of the research, and suggests areas of future work.

Chapter 2

Background

The intent of this thesis is to develop an algorithm and to demonstrate its effectiveness in a realistic UAV mission environment. To this end the Charles Stark Draper Laboratory's Chayton program was selected to define both mission requirements and an overall autonomous planning architecture in which the developed algorithms could potentially reside. This chapter provides background for an autonomous rotorcraft mission based on the Chayton program in order to define algorithm requirements. Furthermore, the specific threats suspected to be encountered must be identified in order to define the specific threat response behaviors needed to enhance mission safety.

A review of threat response algorithms as well as promising trajectory generation algorithms is also provided in order to compare and contrast the pros and cons of each solution method. This comparison leads to the selection genetic algorithms to be used for threat responsive trajectory generation.

2.1 Mission overview

The Chayton mission consists of a set of activity points that are to be visited over the course of a mission by a team (or teams) of autonomous rotorcraft (AR). These activity points are either reconnaissance points or strike points. Reconnaissance points, as the name suggests, are points at which the vehicle is intended to arrive at and operate

certain sensors in order to gather the required information. Likewise, strike points are points at which certain targets are to be destroyed, requiring the vehicle to acquire the target and fire the weapon designated for the target.

Figure 2-1 shows a typical scenario an individual AR would be faced with. The diamonds represent reconnaissance points, the circles strike points, and the x's mark the positions of known threats. The circles around the threats are indicative of the estimated threat range based upon the threat type. Notice the threat in the upper right hand corner of the figure is to be destroyed, indicated by the strike point at which weapons are to be fired from. Although this particular threat does not encompass any other activity points, it may be desirous to eliminate it to clear the path for following missions.

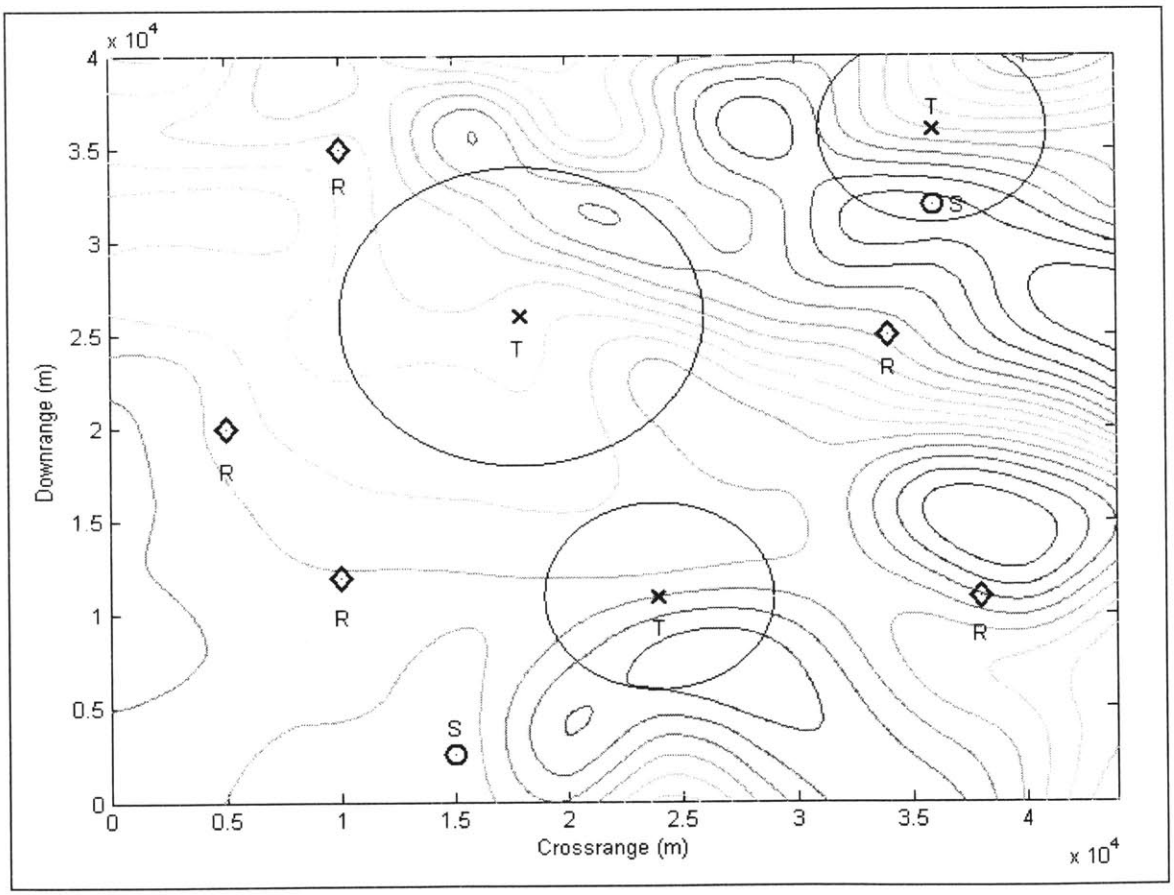

Figure 2-1 Example AR mission scenario

2.1.1 Threat capabilities

The discussion thus far has assumed some *a priori* knowledge of both threat locations and threat capabilities. One frequently encountered military threat that poses a

great risk to a slow flying (relatively), low-altitude AR is the man portable anti-aircraft device (MANPAD). The MANPAD is designed specifically in order to combat low flying vehicles seeking to evade radar detection through terrain masking. In addition, rocket propelled grenades (unguided rockets) and small arms fire pose risk as well [9]. While threats such as anti-aircraft fire and armored vehicles also create risk for the AR, the ease of portability, high proliferation, and difficulty in detection make infantry based threats the more deadly foes.

One such weapon, the SA-14 Grail, is a common MANPAD that has been in service since 1978 (Figure 2-2). The SA-14 is a solid motor propelled missile with a passive infrared seeker head which flies at approximately Mach 1.75, with a range between 4,500-6,000 meters. MANPADs typically have lower accuracy than larger radar enabled surface to air missiles (SAM), with a probability of kill for MANPADs on the order of 30-40% per shot. The infrared (IR) seeker performance is degraded by aircraft orientation, which affects the IR signature of the vehicle, as well as ground heat in low-altitude flight. The launch delay for a single weapon system is 35-40 seconds, including reload, target acquisition, and firing time [10].

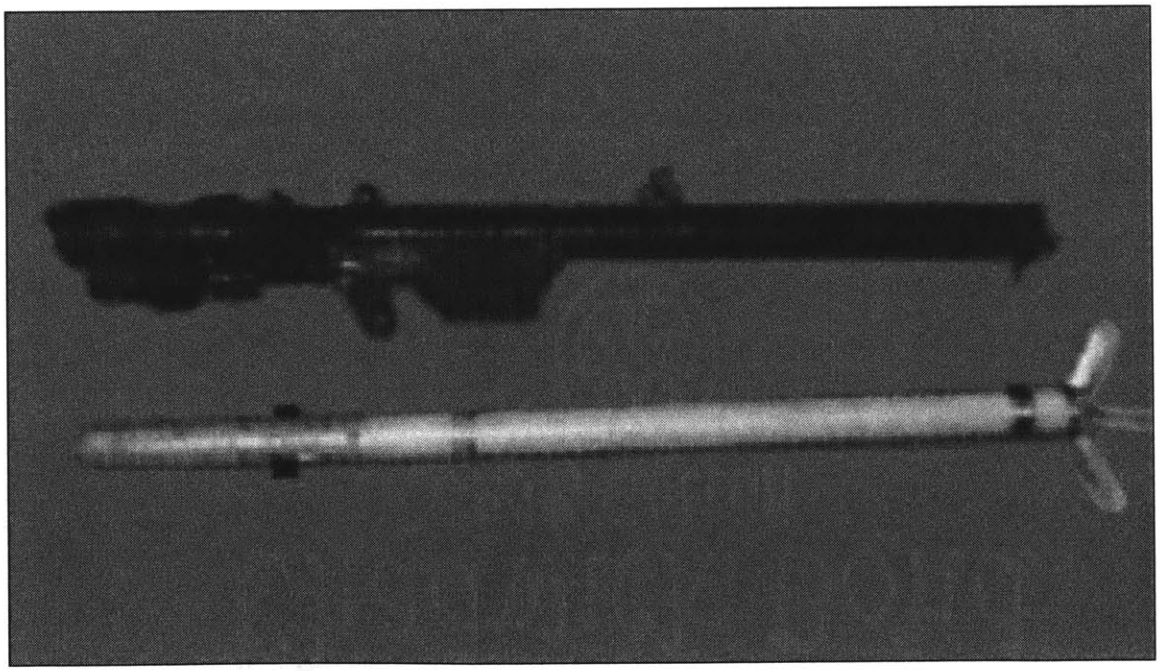

Figure 2-2 SA-14 shoulder launched missile [10]

Although the single shot kill probability is lower for a MANPAD than a radar guided SAM, the MANPAD lethality is effectively increased due to the difficulty in detecting MANPAD locations. Because the MANPAD uses IR tracking as opposed to radar, typically the initial threat detection occurs *after* the first rocket has been fired. Once a MANPAD location has been identified, the most effective strategy to reduce risk is to break line-of-sight (LOS) through terrain masking. Firing the MANPAD requires the human operator to first visually identify the vehicle, and then to point the rocket at the vehicle in order to obtain IR lock. If the MANPAD operator is unable to visually acquire the vehicle the rocket cannot be launched. The second step in risk reduction is to flee the threat radius defined by the range of the MANPAD. While it is preferable to break LOS first if possible, the threat range should be evacuated whether LOS can be broken or not. This strategy of distancing the vehicle from the threat while attempting to reduce LOS exposure is even more effective against small arms fire and rocket propelled grenades due to the fact that both of these weapons are simply bore-sighted when fired.

2.2 Threat evasive response algorithm

Consider once again the example mission shown in Figure 2-1. Let us assume an activity point sequencing planner (either autonomous or human) has determined the order of execution of the activity points, and that the strike point in the upper left hand corner of the figure is to be visited last. Once the order of execution has been established, a route planner would be called to define a safe path for the AR to travel between the activity points. Many algorithms for route planning around threat areas exist in the literature (see Section 2.3.1), and therefore the topic of route planning will not be discussed in this thesis. For our purposes it will be assumed that one such routing algorithm is called to define a safe path for the AR to travel between the activity points which avoids the known threats. Figure 2-3 shows the ordering of the activity points as determined by the activity sequencer, and the general path which the route planner might select in order to evade the threats at the most basic level.

Now consider the scenario illustrated in Figure 2-4. The AR is traveling along the path determined by the route planner when a MANPAD threat is detected in close proximity to the vehicle. The route planner's threat avoidance mechanism is no longer the

best means to maximize survivability since the vehicle is well within the threat's footprint.

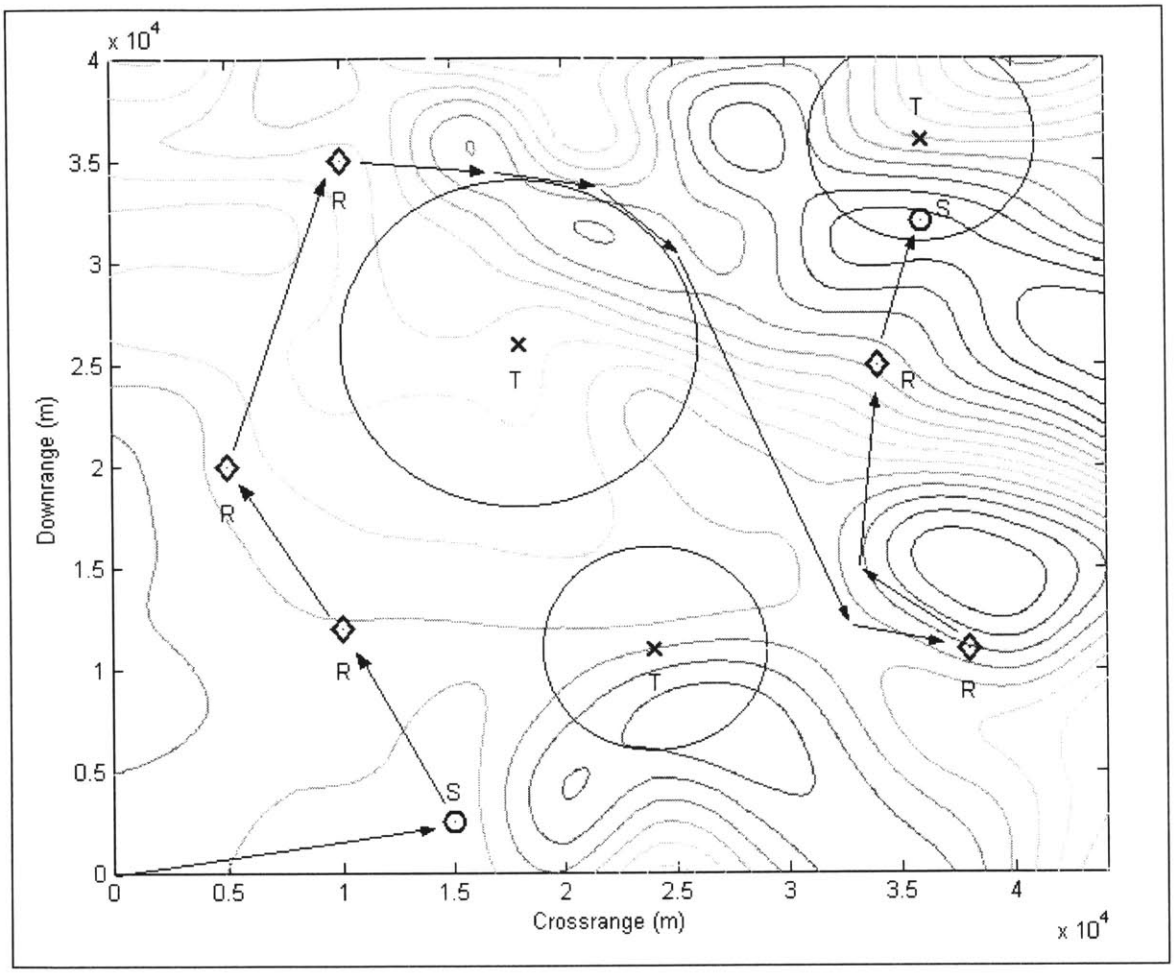

Figure 2-3 Initial UCAR plan for example mission

In this instance, the threat evasive response algorithm (TERA) temporarily assumes control of the vehicle, and the mission objectives are put on hold. Given the current vehicle location, along with the threat type and location, the TERA planner must determine a vehicle trajectory (including position, altitude, and velocity) that takes the vehicle out of the threat's range while attempting to maximize survivability. Because of the low-altitude aspect of the AR mission, TERA must also provide trajectories that prevent the vehicle from crashing into the terrain; however, the terrain also provides the most significant means for enhancing survivability as well. Since the ability of a MANPAD to fire upon an AR depends on the operator's ability to visually detect and

track the AR, breaking LOS between the threat and the AR will negate the ability of the threat to fire upon the vehicle. Because of this, it is imperative that TERA take advantage of terrain masking when planning trajectories. The TERA planner completes its planning objective when the vehicle has safely exited the pop-up threat range and notified the activity sequencing planner that a plan deviation has occurred. Once the threat has been evaded the sequencing planner must assess whether a re-plan of the overall mission is required.

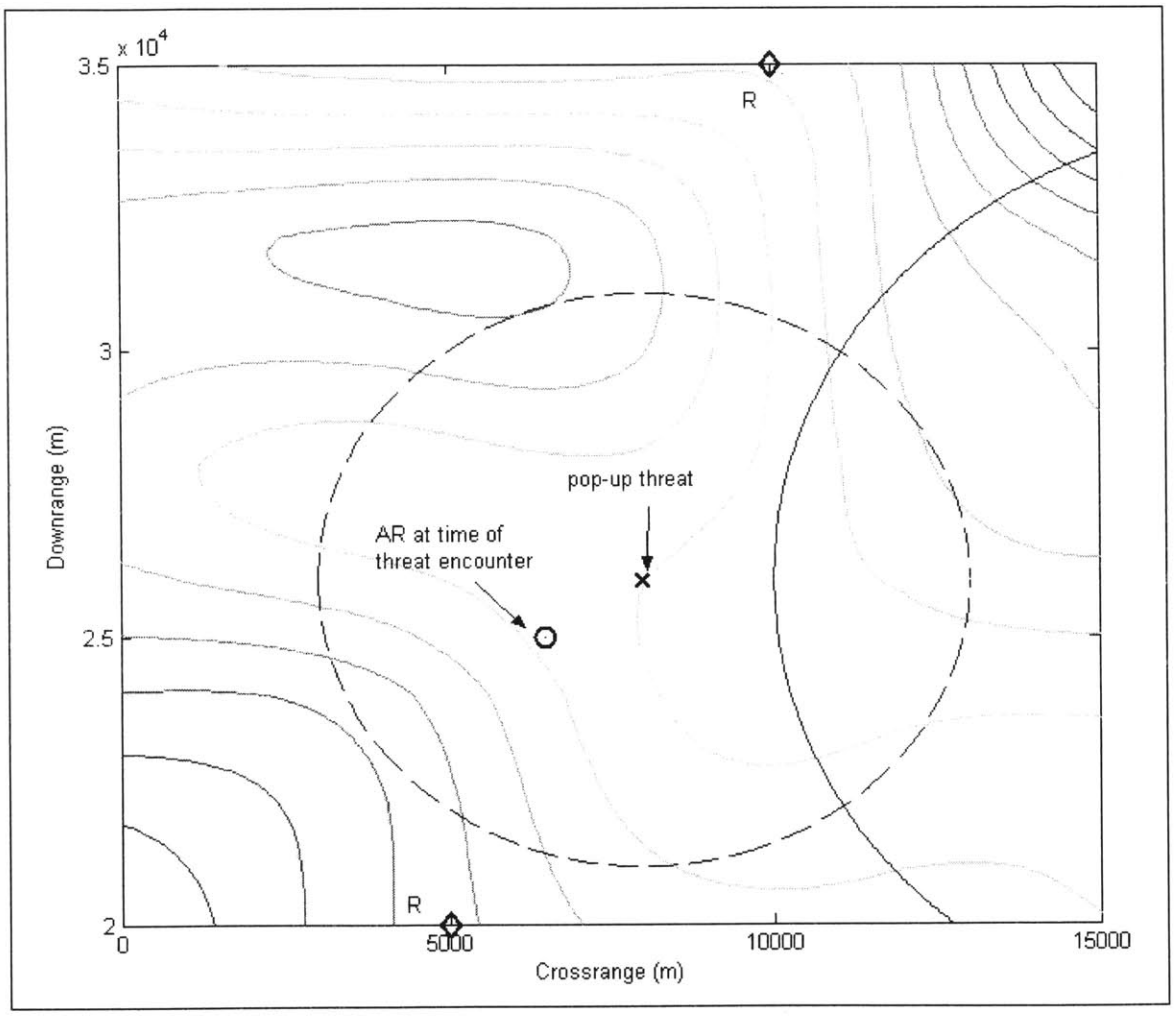

Figure 2-4 Pop-up threat encounter during mission execution

2.3 Discussion of algorithms

A brief discussion on previous work in both threat response as well as trajectory generation algorithms is presented in this section in order to try and identify a candidate

TERA solution method. Of key interest in different solutions to threat response planning is the characterization of the threat environment and the fidelity of the resultant vehicle plans. Likewise, many achievements have been made recently towards trajectory generation for higher dimensional problems, therefore some review of existing technology is warranted in light of the demands of low altitude AR operation.

2.3.1 Threat response algorithms

Threat response algorithms have periodically been of interest in the research community, the first applications to autonomous aerial vehicles appearing in the literature in the mid 1980's. These different algorithms come in a variety of flavors, but for the most part fit into the categories of graph search, geometric, or optimal. Each of these classes of algorithms will be analyzed with respect to the applicability to the problem at hand.

2.3.1.1 Graph search algorithms

The first proposed algorithms for threat avoidance made use of graph search methods exclusively. The Dynapath algorithm [11], being the first introduced in the literature, was interestingly enough one of the few to make use of terrain to reduce threat exposure. The effect of terrain masking was implicitly included in cost minimization through the assumption that by penalizing altitude above sea level one could increase terrain masking by virtue of flying low in general. The overall objective of the algorithm was to determine a 2D horizontal trajectory following a straight line connecting waypoints, allowing for cross-track deviation in the interest of flying in low altitude regions. The possible flight maneuvers were characterized by fixed time step commands consisting of a discrete set of turn rates, which were in turn used to build a tree of fixed horizon length and constrained to a maximum cross-track deviation (Figure 2-5). This tree was searched using Dijkstra's algorithm and appropriate pruning to respect flight corridor constraints, and finally the vertical aspect of the trajectory was filled in using conventional terrain following algorithms for fighter aircraft.

Stanley and Bate proposed a route planner for a fighter aircraft based on an A* search approach [12]. In their algorithm, a coarse, 2 km cost grid is searched using A* to

determine a rough cut path. The cost grid for this step was determined at each point as a function of known threat exposure in an area, as well as a cost scaled by the probability of an unknown threats being in the region. The second planning stage followed by searching a tree via A* formed within a set corridor of the original path using 1 second maneuvers which allowed increasing, decreasing, or performing no change in turn rate. During this stage the cost map is augmented by a penalty for the length of time exposure in direct LOS of a known threat. One of the key advantages with this algorithm is the explicit dependence on LOS, a factor already established as being of major importance for threat evasion.

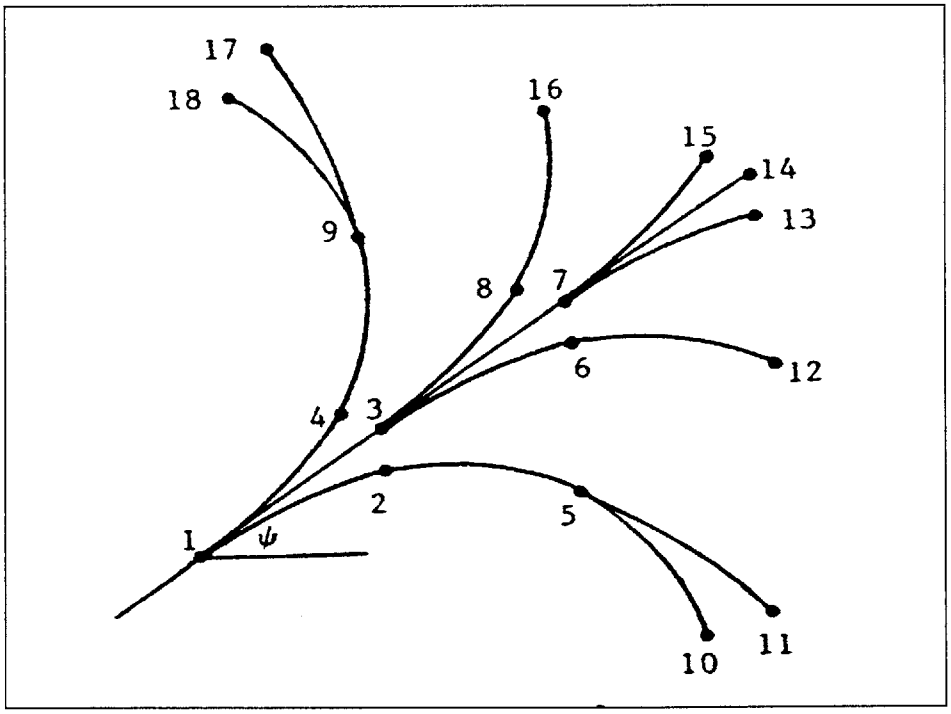

Figure 2-5 Dynapath tree generation [11]

One of the key drawbacks in using a graph search based algorithm is the fact that computational complexity explodes as trajectory resolution increases. Using A* over a Dijkstra search can reduce run time by pruning intelligently, however if the tree were expanded to include vertical maneuvers as well as velocity, the solution quickly becomes intractable as the search horizon or resolution in maneuvers increases.

In addition, another issue exists which makes this approach less appealing for our needs. Bate and Stanley noted (also mentioned in other research as well [13]), the

determination of the optimistic cost to go heuristic needed for A* search is quite difficult. Both [12] and [13] attempted the use of straight distance to the goal to generate the cost, the major issue there being what happens if the straight line to the goal point passes directly through a threat region? Another method attempted to remedy this problem was to estimate the optimistic cost to go through the use of a backwards A* search for each cost to go estimate. The conclusion reached was that straight line cost gave unrealistic performance, however the more advanced backwards A* cost to go estimation increased computational complexity unduly.

2.3.1.2 Geometric based algorithms

Several geometric based algorithms have been proposed in the literature. One common approach is to construct Voronoi polygons about known threat emplacements to generate a graph of possible routes through a region. Algorithms making use of Voronoi paths typically include further processing to make the resulting paths dynamically feasible. For example, Judd and McLain present a two-dimensional, constant velocity, horizontal plane planner which constructs Voronoi polygons around known threats, searches the resulting graph using Dijkstra's algorithm to find the lowest cost path, and finally fits cubic splines between Voronoi edges in order to respect turning rate constraints of the vehicle [14]. In this case the cost is defined as a sum of edge costs which are proportional to $1/\text{range}^4$, evaluated and summed across all edges. Similarly, in [15] a Voronoi graph is determined, searched using Dijkstra's algorithm, and then made feasible by adding fillets to corners. In this case velocity is assumed constant along the path, however the magnitude is selected in order to meet time of arrival constraints. Pop-up threats are included to force re-planning, consisting of rebuilding the Voronoi diagram to account for the new threat.

Bortoff proposed an interesting variant of these algorithms in which the resulting Voronoi path is used as an initial condition to a dynamic simulation intended to refine the resultant path [16]. The path is modeled as a chain of point-masses connected in series with springs and dashpots. Each threat is described as a repulsive force, and a simulation is run to find the minimal energy state of the system, returning the final two-dimensional path.

Another method is presented in [17] which constructs paths through a threat rich region by constructing paths that tangentially connect circles used to describe each threat's range. In order to move from a point 'A' to a point 'B' the vehicle can move along the circumference of a threat circle, or along straight line paths connecting circles. As new threats are introduced the re-planning activity consists of searching for alternate paths around any given threat that intersects the previous path solution.

One drawback to using one of the algorithms discussed is that they can be ill-defined in regions of sparse threats. If a vehicle operating in a region with no immediate threat coverage was to encounter a pop-up threat while within the threat's range, the resultant Voronoi polygon would contain no edges, and hence provide no insight into evading the threat. In addition, these methods are inherently two-dimensional, which is not an ideal starting point for the four dimensional trajectory planning problem we wish to solve. While altitude could be added in a secondary step (as in the Dynapath algorithm), this decoupling reduces the power of the algorithm to exploit factors which are extremely three-dimensional in nature (such as LOS). Perhaps the greatest weakness is that the algorithms discussed are not easily adapted to provide detailed planning while within a threat sphere. While it may be possible to extend algorithms of this type in order to generate three dimensional search graphs using the edges of threat region, as the number of edges within the graph grows it becomes intractable to perform an optimal search over the graph.

Figure 2-6 illustrates a typical threat scenario used in a geometric formulation. The scale of the solution shown is obviously of sufficiently low resolution that even were the method used for vehicle routing a high resolution trajectory planner would still be required to determine the near term trajectory response in order to increase survivability.

2.3.1.3 Optimal algorithms

Optimal control algorithms have likewise seen a good deal of interest in trajectory optimization for threat response [18, 19, 20, 21, 22, 23]. The greatest advantage of these algorithms is that a dynamic model of the vehicle is used as a constraint for the minimization of a cost functional. The maneuvering limits of the vehicle can be included by imposing further state and control constraints during the optimization, resulting in

trajectories which are flyable by the vehicle while providing means of proving optimality in results.

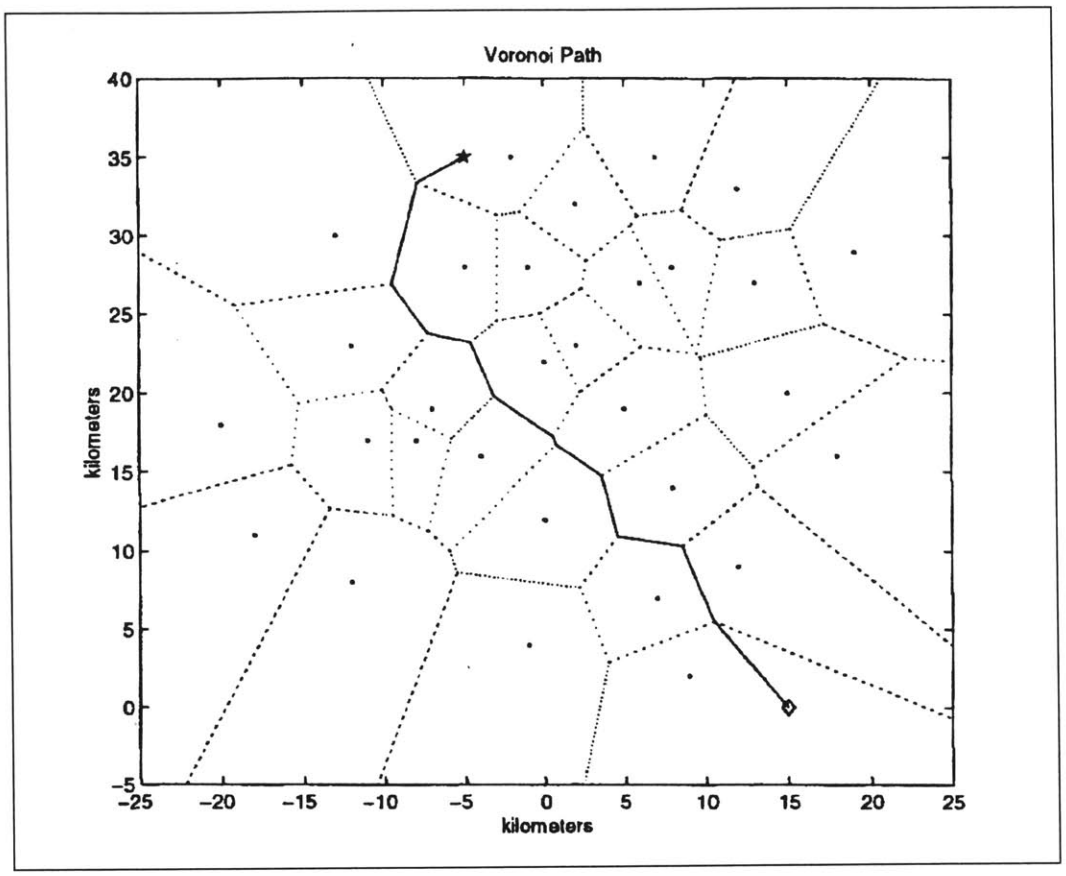

Figure 2-6 Typical threat environment for Voronoi based planning [14]

While this appears to offer the greatest potential for providing the feasible four-dimensional trajectories we require (computational complexity due to four states aside), one significant hurdle remains. Each of these algorithms requires an analytically describable and continuous cost functional in order to proceed with minimization. When operating in the low altitude regime, LOS between the vehicle and threats becomes a major contributor in the cost of a specific trajectory, and at any given point LOS is a binary operator. Line-of-sight cannot be described in a continuous fashion, making any cost function including LOS effects is inherently discrete and discontinuous in nature, making optimal solutions impractical.

2.3.2 Kinodynamic planning

Having explored the options available in the literature for threat avoidance and finding none that specifically address the requirements for the TERA planner, we must look to other areas of research in trajectory generation. The area of kinodynamic planning has received much attention in the literature recently. The term "kinodynamic planning" was coined to describe path planning that takes into account *dynamic constraints* in addition to *kinematic constraints*. Traditionally, robot path planning has taken place in the configuration space, a representation used to describe the locations of physical obstacles in the operating environment [24]. By far the most common path planning problem solved in the configuration space is one of determining a collision free path through a complicated maze or set of obstacles. These paths typically are concerned with circumventing obstacles, and less with observing the dynamic capability of the vehicle.

Kinodynamic planning on the other hand is performed in the state space (described by both positions and velocities) and therefore is capable of incorporating dynamic vehicle constraints as well as performing trajectory planning over velocity dimensions. This is particularly useful to a potential threat response planner as the probability of attrition due to a threat is generally a function of exposure time. Because of this, a threat response planner should be able to return solutions involving control over velocities, or alternatively, over time. Of particular interest are algorithms capable of efficiently generating results, as the kinodynamic planning problem is thought to be at least as difficult as the Mover's Problem which is known to be PSPACE-hard [25].

Frazzoli presents a dynamic programming based motion planning algorithm which is capable of real-time trajectory generation [26]. This algorithm ensures dynamic feasibility by generating a set of maneuvers and trim conditions using a dynamic model of the vehicle. The algorithm then "stitches" these maneuvers and trim conditions together in order to build a trajectory. By generating each trim and maneuver based on the dynamics of the vehicle, the resulting trajectories are guaranteed to be feasible. This method provides a bottom up approach to trajectory planning, as opposed to the traditional top down approach of imposing simplified constraints in an attempt to generate solutions that can hopefully be tracked by the flight control system. The algorithm is based on the fact that many vehicles are invariant to certain group

translations, in particular, disregarding effects of atmospheric density, the time required for a certain helicopter maneuver is invariant to the initial (x, y, z) position. Therefore, a library of maneuvers which connect various trim conditions, their time costs, and a cost-to-go matrix can all be computed *a priori*, and on-line trajectory generation can be performed through application of Bellman's "principle of optimality". The result is a time-optimized, dynamically feasible trajectory which can be computed in real-time, and is capable of capitalizing on the agility of the vehicle.

The method devised by Frazzoli provides an excellent means to generate real-time trajectories to the full extent of the vehicle's capability; however, the base assumption of invariance to group translation in the configuration space can only be made for the solution of minimum time trajectories. When threat range and LOS are factored into the trajectory cost calculation, maneuver cost become entirely dependent upon position. Because of this, Frazzoli's algorithm is not a viable option for threat evasion.

Much of the research as of late in kinodynamic planning has been in randomized planners, due to the success of Probabilistic Roadmap (PRM) techniques for high dimensional planning in configuration space [27]. These techniques typically involve the construction of a roadmap of free paths through random sampling of the configuration space. This is accomplished through connection of randomly placed nodes, or "milestones". The resulting graph, or probabilistic roadmap, is then searched for an obstacle free path.

The concept of PRM's was extended to the state space by LaValle and Kuffner in a method named randomly-exploring random trees (RRT) [28]. Since then several variations of the RRT algorithm have been introduced [26, 29] in attempts to generate more probabilistically complete algorithms. Essentially, RRTs form a roadmap in the state space by placing milestones with varying levels of randomness (depending on the version). In [28] the tree is formed by repeatedly placing a random milestone in the state space, picking the node on the existing tree that is closest in some sense (Euclidean distance, for example) to the milestone, and applying the control that moves the state towards the milestone for some fixed length. In [26] the tree is formed by first selecting a random milestone in the state space. An optimal control is then applied beginning at each node in the existing tree in an attempt to connect a node to the milestone. Once a feasible

solution to the endpoint is found, the emphasis changes to minimizing the cost of the solution. By introducing intermediate milestones and selectively pruning the tree based on cost, the algorithm is able to refine the solution in order to reduce the solution cost.

The primary focus of the RRT-like algorithms is to find a clear path through an obstacle field while incorporating dynamic constraints of the vehicle. However, the typical battlefield mission will not be flown through a field of floating obstacles, hence the uniform exploration of the space is not as important for TERA. The actual goal is to direct the search toward minimizing the cost of threat exposure in a setting where the terrain provides the only kinematic constraint (the constraint being that trajectories cannot fly underground). An RRT based algorithm for threat response could possibly be created through the introduction of heuristics to guide the search towards the required goal; however, this would serve to reduce the exploratory ability which is the fundamental purpose of the RRT. In addition, the quality of the solution would certainly depend on the connectivity of the search tree, which could require an exceedingly large number of milestones since no goal state exists for threat evasion (see Section 2.2).

2.3.3 Genetic algorithms

The success of randomized kinodynamic algorithms for trajectory generation in high dimensional space suggests that searches including a random element are preferable to enumeration. Genetic algorithms (GA) are another class of algorithms that use a randomized directed search to generate solutions. Genetic algorithms have proven effective in generating configuration space [30, 31, 32], as well as kinodynamic solutions [33] for vehicle routing. In [30], the authors develop a robot path planner that proves adept at determining collision-free paths through complex mazes. The authors of [33] create a genetic algorithm for three-dimensional (horizontal plane plus velocity) UAV motion planning in environments with uncertain obstacle locations.

Genetic algorithms have several characteristics that make them favorable for trajectory generation for threat response. One of the significant differences between GAs and other optimization schemes is that GAs do not take advantage of any known structure of the cost function undergoing minimization. Instead, the GA simply makes use of the cost *value* of each potential solution, which can come from a cost function of any form.

Many optimization methods require the use of derivatives of the cost function in order to develop a solution. The fact that GAs do not use cost function knowledge such as derivates turns out to be quite beneficial in the threat response planning problem. Optimization methods that require use of derivative information typically require the cost function to be finitely differentiable to a certain order, which restricts the form of the cost function. In threat response, if LOS to a threat is broken the cost of the threat effectively becomes zero. This binary cost multiplier makes the perspective cost function non-linear and highly discontinuous. Furthermore, the effect of LOS cannot easily be expressed analytically, as it is a function of terrain which is constantly changing as the vehicle travels. The effect of LOS is easy to determine, however, if a trajectory is presented and the cost is merely to be evaluated. This is precisely the way genetic algorithms operate, making them convenient for the problem at hand.

Along with the cost function restrictions gradient methods impose, unless the cost function is known to be globally convex (or concave), the use of derivatives inevitably introduces the potential for finding local minima (or maxima). While the possibility exists that a GA will find a local minimum of a problem, there exist processes in their operation which assist in the search for the global optimal solution. Fundamentally, a GA is a directed randomized search that works a set of complete candidate solutions in parallel. The intent of working with a set of solutions is that the solution space can be more readily searched out in order to allow the GA to localize onto a likely solution for the global optimum.

Some graph based search algorithms are capable of determining the globally optimal solution to the resolution of their discretization. Dynamic programming, for example, pre-computes the optimal controls and costs to go between various points in the state space which are then used to create an online feedback controller guaranteed to find the optimal solution. The A* algorithm performs a similar process to dynamic programming, however cost to go is estimated via a heuristic. Under certain conditions upon the heuristic, A* is also guaranteed to generate the globally optimal solution. However, due to the enumerative nature of both A* and dynamic programming, they suffer from the "curse of dimensionality", i.e. they quickly become too computationally intensive if the dimension of the search space becomes large. Genetic algorithms, on the

other hand, require no off-line computation or knowledge about behavior of threat risk throughout the state space. This is an advantage in a dynamic threat environment in which changes in threat knowledge have the potential to invalidate previous risk information. The lack of offline computation allows the GA to readily adapt to any changes in the threat environment. Additionally, because GAs are not enumerative in nature, they do not suffer the "curse of dimensionality".

Ultimately, the desired result of a threat response planner is a system which can trade off between "good enough" and "fast enough". Here the old adage, "a bird in hand is worth two in the bush" applies in spades. A non-optimal solution returned in time for the vehicle to react intelligently to the threat is more desirable than an optimal solution returned too late. Genetic algorithms inherently offer a tradeoff between efficacy and efficiency. As will be discussed in Chapter 3, GAs begin with candidate solutions and work iteratively to reduce the cost of the best solution. This gives the user the control to balance the quality of the solution with timeliness. Should the algorithm need to be stopped prematurely for any reason a solution will still be provided (i.e., it is an *any-time* algorithm); however, the more time that can be afforded to the algorithm, the better the solution will be.

2.3.4 Comparison of algorithms

Table 2-1 shows a comparison chart of all of the classes of algorithms discussed in Section 2.3. The table compares the algorithms based on the ability to include a high number of states, whether the results are deterministic, whether the algorithm finds the guaranteed optimal solution, whether the algorithm can be stopped at any time and still yield a solution, the ability of the algorithm to handle LOS in the cost function, the ability to include dynamic constraints, and whether the algorithm is an optimization in the sense that the goal is to minimize a cost function.

It is clear that no one algorithm outperforms the other for all circumstances, which is to be expected. The table does, however, provide insight into which algorithm offers the most promise for TERA. Based on the requirements of TERA we know that a potential algorithm must be able to (1) solve a four-dimensional trajectory generation problem, (2) include LOS in the cost analysis, and (3) include dynamic constraints. Based

on these minimal requirements the only two classes of algorithms suited for TERA are either the RRT-like algorithms or genetic algorithms. However, the fact that the RRT-like algorithms are intended for uniform exploration of space in order to find a solution, as opposed to optimization of a cost function, makes them a weak contender for TERA. Furthermore, once the any-time ability of genetic algorithms is factored in, the decision to develop TERA using GAs becomes clear.

Table 2-1 Comparison chart of algorithms discussed

Attributes	TERA requirements	Graph search	Geometric	Optimal	RRT	GA
easily extendable to high number of states	R			X	X	X
deterministic	D	X	X	X		
optimal	D	X	X	X		
can yield feasible solution at any time	D					X
can handle discontinuous cost function (for LOS)	R	X			X	X
ability to incorporate maneuvering limits	R			X	X	X
goal of algorithm to minimize cost	R	X	X	X		X

R = required
D = desirable

Chapter 3

Algorithm Implementation

This chapter presents a genetic algorithm design for solving the threat evasion response problem discussed in Chapter 2. The first part of the chapter provides the necessary background in GA operation. The remainder of the chapter is dedicated to discussing the specific GA design for TERA application, and in particular how dynamic feasibility is ensured during the trajectory generation process.

3.1 Overview of genetic algorithms

Genetic algorithms are surprisingly simple in their general mechanics of operation. In short, GA's work with a set of candidate solutions to the optimization problem in an iterative process designed to refine the solutions in the direction of lowered cost. The GA analogy is of a simplistic evolutionary survival-of-the-fittest process. Each candidate solution, or chromosome, is a part of the larger set of solutions maintained during each iteration. As per the analogy, the set of solutions is known as the population, while each successive step in the iteration process is a generation. Those chromosomes that stand out from the population receive a higher chance of having some or all of their "genetic material" reproduced in the following generation. Those population members that perform poorly with respect to the optimization rubric are less likely to be used in the

next generation, and hence the population has a tendency to better itself as the number of generations increase.

The chromosomes as defined above are each representative of a possible solution to the optimization. As such, they must contain a method of fully describing a solution to the problem at hand. In general, a chromosome is comprised of a string of numerical or binary values which can be interpreted as a complete solution. For example, to minimize the cost function $f(x) = (x-5)^2$ in the interval $0 \leq x \leq 50$ the user may choose to represent each value of x by a fixed bit length binary string. Indeed, much flexibility exists in the way chromosomes are represented, which is a direct result of the problem being solved. Chromosome representation will be discussed in greater detail in Section 3.1.1.

As in nature, there exists several means of which good population members can be propagated to the next generation. Genetic algorithms make use of reproduction, crossover, and mutation as the primary mechanisms for population refinement during each generation. In reproduction a chromosome is simply copied directly into the population of the next generation. In crossover two parent chromosomes are combined to produce an offspring. This is achieved by copying a portion of one chromosome and a portion of the other chromosome and concatenating these portions to produce a new chromosome, typically of the same length as the two parent chromosomes (the crossover operation will be discussed more in Section 3.1.3). The mutation operator selects a single element, or gene, of a chromosome and changes the gene value randomly with some small probability. While the reproduction and crossover operators are intended to propagate "good" genetic material into the next generation in an attempt to better the chromosomes, the mutation operator is generally not used in a deliberate action to increase a chromosomes value. Rather, mutation offers the possibility of boosting performance by shaking the algorithm out of a local minimum.

So far it has been assumed that some chromosomes are better than others, and therefore deserve to be reproduced in some fashion in the next generation. As in any optimization routine, some form of cost function exists which is to be minimized over. The GA is no exception to this, and therefore each chromosome's cost, or fitness, is evaluated via a user-defined cost function. In the example used above, the cost function is $f(x) = (x-5)^2$, for which a binary chromosome string representing the value 6 has a better

fitness than a chromosome representing the value 42. Notice here that the cost function requires the input of a number rather than a binary string. This illustrates the concept of input space and output space, used often in genetic algorithms. The input space refers to the format in which the chromosome is represented, in this case as a binary string, while the output space refers to the format of the desired output. When these are different some decoding is necessary to map the chromosome to the domain of the cost function. In the example given the user may want to map a given binary chromosome to a real number to use in the cost function, instead of using binary arithmetic.

In order to describe a full GA implementation it makes sense to first describe in more detail some of the specific mechanisms used. Once this has been done we will discuss a simple GA in order to illustrate the application of the algorithm.

3.1.1 Representation

One of the main advantages in using GA's is the flexibility in representation of the solutions in the chromosome; however, determining what representation to use can often be the most difficult part in developing a GA. Each gene can be comprised of a single value, or of a set of multiple values that when taken together are representative of an element of the solution. Furthermore, a gene element can be expressed as a number (or character), or as a fixed length binary string. One of the drawbacks to binary representation, however, is that it takes away intuition of some of the GA operators. Binary representation was favored in the early days of GA research, however many researchers have turned to non-binary representation because of the insight gained in intuition without loss of generality of the GA.

As a concrete example of representation, consider the path planner presented by Xiao *et al* in [30]. The goal of the GA was to generate a path for a ground based mobile robot from given points 'A' to 'B' while avoiding obstacles. Each chromosome contains a path represented as a set of waypoints leading from start to finish. Each gene consists of three values describing a waypoint in the path. The first two values are the x and y locations of the waypoint, while the third value conveys whether the waypoint or the path leading from the waypoint intersects an obstacle. Furthermore, the order of the genes within the chromosome depicts the order in which the waypoints are visited. Figure 3-1

shows two possible chromosomes. Note that even though both chromosomes contain the same waypoints, the solution path is different due to the different orderings. This illustrates another tool the GA designer has available in that the location of the gene within the chromosome can carry significance. This is very prevalent in problems where the chromosome describes a temporal sequence in which case gene location is indicative of time ordering of the solution.

Yet another tool the GA developer has available for chromosome representation is used by Xiao, this being variable length chromosomes. In the path planning example this means the GA has control over not only the waypoint locations and ordering, but also the number of waypoints visited. Allowing the chromosome length to vary effectively increases the search space of the algorithms, and typically some limit must be set on the overall length of the chromosome. Variable length chromosomes are not as common in application as fixed length, however length remains as an option in representation.

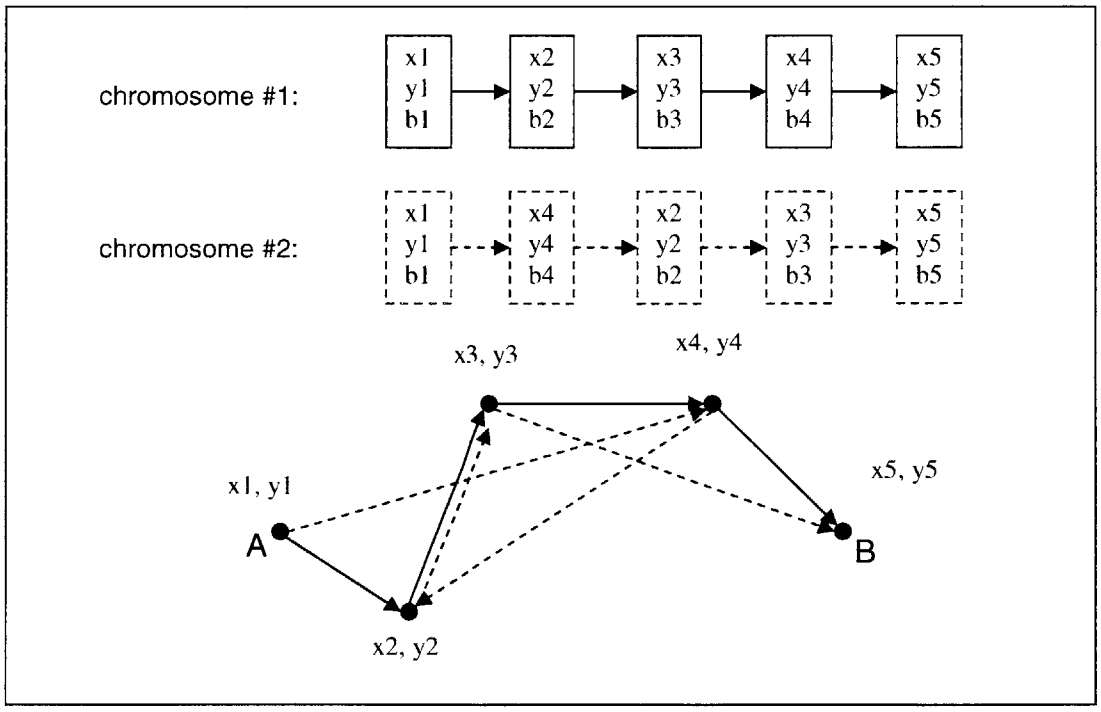

Figure 3-1 Chromosome representation for path planning

3.1.2 Selection

In order to ensure "better" chromosomes are reproduced in some fashion from one generation to the next, higher performing population members must be shown

preferential treatment during crossover and reproduction. Many options exist for selection of individuals based on fitness, each providing slightly different probabilistic performance.

The most common method is via roulette wheel selection. In roulette selection each chromosome receives a proportional number of slots on the roulette wheel to the percentage of total population cost possessed by that member (see Figure 3-2). A random number is then drawn simulating a roll of the roulette wheel, and a population member is chosen based on the value of the random number. In this way chromosomes with higher fitness will be selected with probability commensurate with their relative performance. In the case of minimization, the reciprocal of the chromosome cost is used to generate the roulette wheel so that chromosomes with low fitness have a higher chance of selection.

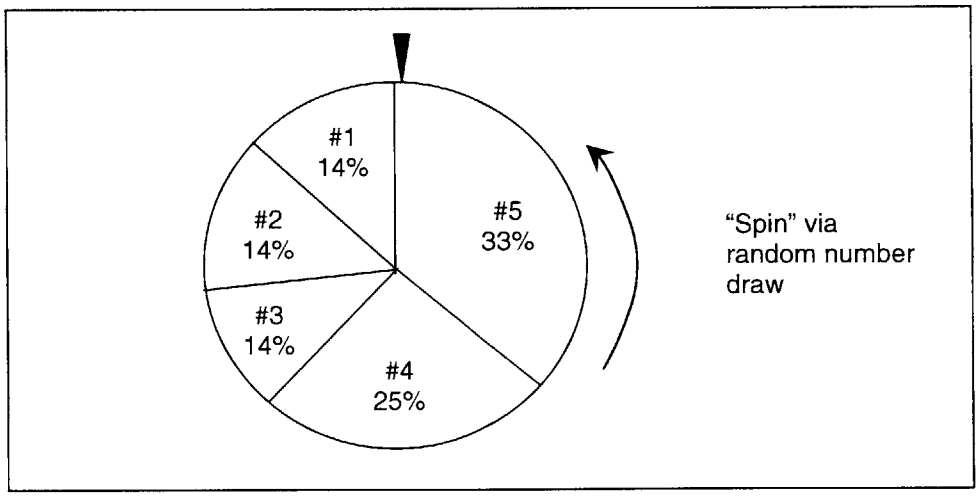

Figure 3-2 Roulette wheel for chromosome selection

Many other selection schemes exist which can be used for fine-tuning the convergence properties of the GA. For information about other selection methods the reader is referred to Goldberg [34].

During initial generations in the GA it is common to have a few chromosomes that vastly outperform others in the population. If the roulette wheel (or any other) selection scheme were used, it is apparent that these individuals would tend to be selected most frequently and therefore dominate the generation, possibly leading quickly to a local minimum. As the number of generations increase a very different behavior becomes possible. As the GA begins to converge the average fitness of the population becomes

close to the best fitness. In this instance the selection process will wander between population members as better solutions don't stand out dramatically.

One way to address this problem is through linear fitness scaling (LFS). In LFS the fitness values are mapped by the linear relation:

$$f^s = mf + b \qquad \text{(Eqn 3-1)}$$

Nominally, the values m and b are selected in order to meet the following requirements:

$$\begin{aligned} f^s_{\max} &= K \cdot f_{avg} \\ f^s_{avg} &= f_{avg} \end{aligned} \qquad \text{(Eqn 3-2)}$$

where K in effect defines the amount of linear spread among the scaled fitness values. It is possible that some value scalings will cause the minimum scaled cost to become negative (Figure 3-3). In this case K must be reduced to prevent negative fitness from occurring. This can easily be accomplished in the implementation of the LFS algorithm by introducing logic to scale K so that $f_{min}{}^s > 0$. The desired effect is achieved by reducing the dominance of outstanding chromosomes early in the algorithm through limiting the ratio of f^s_{max}/f^s_{avg}. Later in the process the scaling increases variance among the population members allowing slightly better chromosomes to stand out, thus enhancing convergence [34].

3.1.3 Crossover

The crossover operation is the backbone of the GA. It allows the recombination of solutions which provides the potential for offspring solutions that perform better than either parent chromosome. Figure 3-4 shows two different types of crossover. In single point crossover two parent chromosomes are selected for mating. A string location is randomly selected as the break point. The genes from the first parent preceding the crossover point are then combined with those genes from the second parent that succeed the crossover point. A similar process is performed using the genes from the second parent first if the operation is to create two offspring (depending on implementation only one offspring may be required). Note the crossover point does not have to be the same for the second offspring.

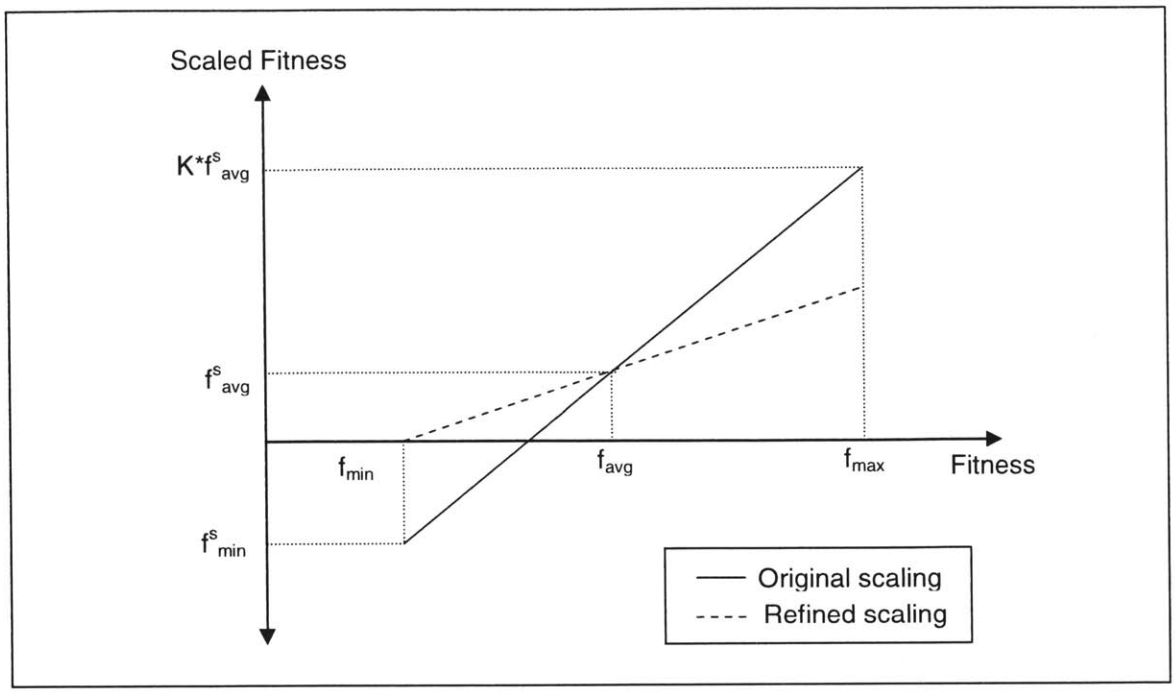

Figure 3-3 Linear fitness scaling with adjustment

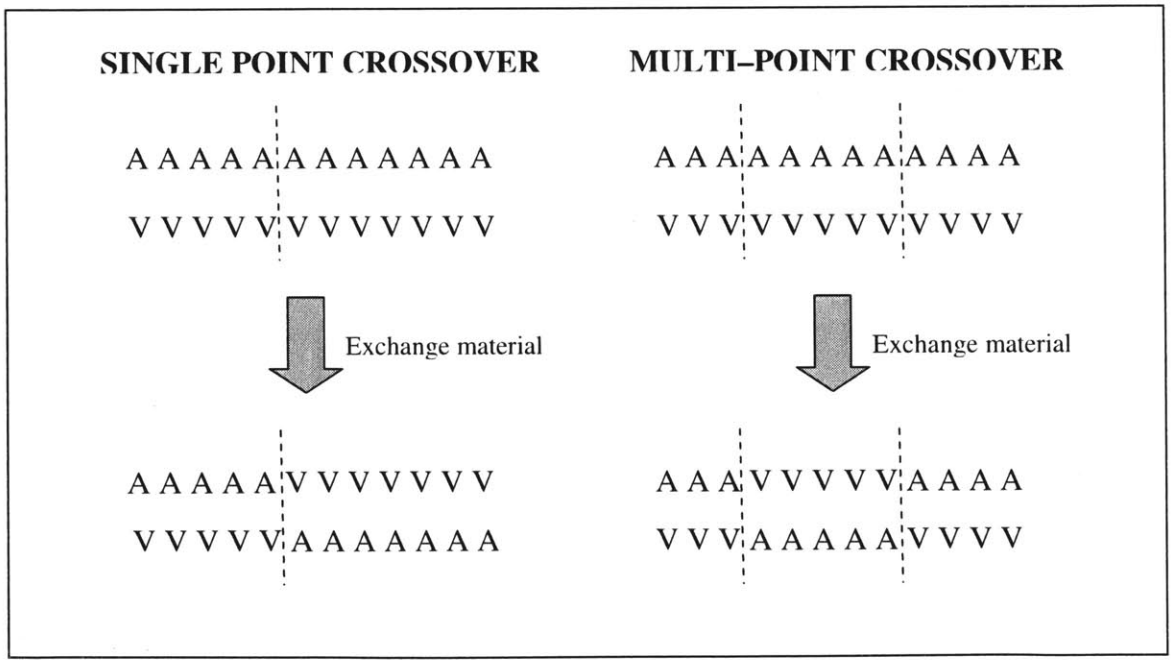

Figure 3-4 Crossover operation

In multi-point crossover q string locations are randomly selected as crossover points. The value for q can be set by the user or can be a random variable. Once the

locations are selected, the genes from the parent chromosomes are swapped as shown in Figure 3-4 for q = 2.

3.1.4 Mutation

During mutation a gene's value is randomly altered to a new value. Figure 3-5 shows an example of a mutation for the path planning problem in which case the third gene in the chromosome is selected for mutation. After selecting the third gene for mutation, within the gene the y coordinate value is randomly selected and replaced with a new, random value y'. In cases like this where multiple optimization parameters are present in each gene, the number of parameters altered in a given mutation remains a tuning factor for the designer.

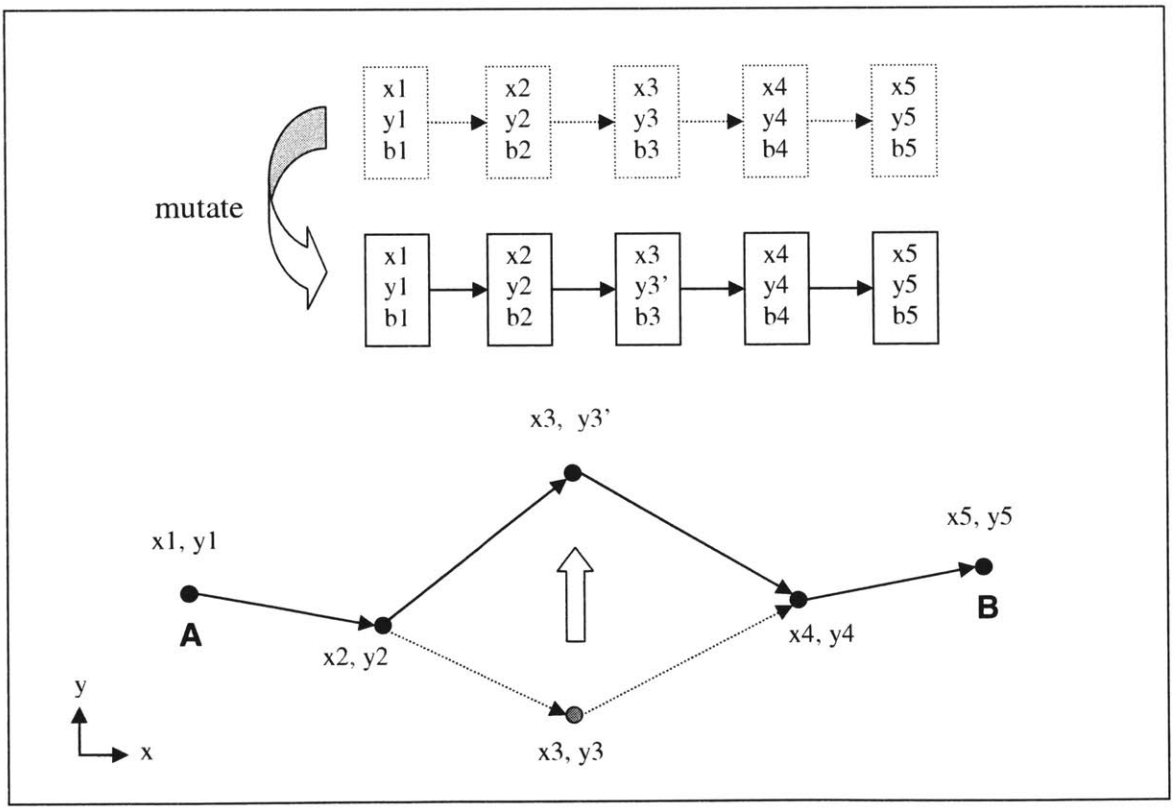

Figure 3-5 Mutation in robot path generation

3.1.5 A simple genetic algorithm

Now that all of the required mechanics have been defined for the GA it is possible to describe a basic implementation method in order to finalize the operation of the GA.

The following pseudo-code describes a complete genetic algorithm using the concepts already discussed. Assuming the chromosome representation has been defined, and given the user inputs of (1) population size, (2) probability of crossover, (3) probability of mutation, and (4) the maximum number of generations the GA proceeds as follows:

initialize the population by randomly seeding each chromosome

gens = 0

while gens < max number of generations {

score each chromosome to determine fitness

Choose 2 chromosomes by roulette wheel. Select a random number, b, in the interval (0, 1). If b ≤ probability of crossover perform crossovers and create 2 new offspring for new population, else copy chromosomes into new population. Repeat this until the new population contains the proper amount of chromosomes.

For each gene in each chromosome generate a random number c in the interval (0, 1). Mutate gene if c ≤ probability of mutation.

gens = gens + 1

end while

return lowest fitness trajectory generated

3.1.6 Constrained optimization

The discussion into GA operation so far has ignored the possibility that constraints exist which limit the feasible solution space. Constraints can be violated during population initialization, crossover, or mutation within the GA. Constraints are typically handled in one of three different fashions in the implementation of a GA.

The first method enforces feasibility by immediately deleting any solution that breaks a constraint. While this method is very fast for removing infeasibility since no time is spent in a repairing process, there are two major drawbacks. The first problem is that if a population size is to be maintained, every time an infeasible chromosome is created and deleted, another one must be made to take its place. Since there is no guarantee that the new chromosome will be feasible, it may be deleted as well. Because of the stochastic nature of the crossover and mutation operations this won't always be the case; however, the possibility of spending a significant effort in repeatedly creating chromosomes to fill a population exists with a finite probability. The other main concern

with this approach is that promising genetic material can be lost when a chromosome with a minor infeasibility, but otherwise good performance, is deleted from the population. This method tends to be ill-advised because of the limiting effect it can have on the creation of new genetic material.

In the second method for constraint handling, solutions are allowed to break constraints, however infeasibility is penalized in the fitness evaluation, making infeasible chromosomes less favored for reproduction. This method is widely used because it removes any restrictions from the crossover and mutation operations. The only drawback is that it can be difficult to create a fitness function for weeding out infeasibility. The fitness of any infeasible chromosome is typically desired to be worse than even the worst feasible solution, however the infeasibility penalty can't be too high or else the infeasible chromosomes will never be selected for reproduction. This often motivates the need for a measure of the degree of "infeasible-ness". In addition, some logic must be included to prevent the algorithm from stopping before a feasible solution is found (that is to say, it is possible *all* solutions are infeasible in a population).

The third method ensures feasibility by limiting the crossover and mutation operations to only produce feasible solutions, or by repairing infeasible solutions generated. An example of how this might be done is presented in [30] for the robot path planning problem, where a mutation operator is introduced that identifies an infeasible waypoint in a chromosome (one that is in an obstacle) and moves the waypoint to a new location outside the perimeter of the obstacle. The intent is to take an infeasible chromosome and make it feasible. The downside of this method is immediately apparent, in that it requires the development of customized crossover and mutation operations that are capable of either producing only feasible solutions, or that are capable of repairing infeasibilities. On the other hand, maintaining feasibility reduces the time spent operating upon and evaluating solution which are infeasible.

3.2 Trajectory generation implementation

The intent of this thesis is to apply the GA principles as discussed in Section 3.1 to the problem of trajectory planning per the TERA requirements discussed in Chapter 2. We define a trajectory as a sequence of three spatial dimensions expressed in an inertial

coordinate system and an associated time at each point. The inertial reference frame is defined as x, y, z, where x is aligned with East, y with North, and z with altitude. All trajectories described in this thesis have a constant Δt spacing associated with consecutive points within the trajectory, however the value of Δt can be varied to the required resolution. Since the trajectory is the desired output of the TERA algorithm, the output space of the GA will be defined as the space spanned by all possible four-dimensional (three spatial plus time) trajectories.

3.2.1 Representation

The first step in creating a GA implementation is to determine the representation of a solution in the input space (that is to say, figure out how to characterize the solutions within the GA). One approach would be to define the input space as the set of positions and velocities at each point in time that define the trajectory, meaning the input space would be the same as the output space. This leads to a waypoint formulation similar to that in [30], with the additional dimensions of altitude and time associated with each waypoint. The drawback with this approach is that a method must be determined to calculate the way the vehicle would traverse from waypoint to waypoint in order to determine whether the commanded waypoints are feasible or not.

An alternative formulation begins by describing the vehicle state in terms of speed (V_n), heading angle (Ψ_n), and flight path angle (γ_n) at a given time t_n, as shown in Figure 3-6. Here speed is defined by the relation:

$$V = \sqrt{v_x^2 + v_y^2 + v_z^2} \qquad \text{(Eqn 3-3)}$$

where v_x, v_y, and v_z are the components of the velocity in each of the inertial coordinate directions x, y, and z, respectively. The velocity heading angle is measured positive counter-clockwise from the x axis within the x-y plane, and the flight path angle is measured from the x-y plane. The following equations provide the relations between heading and flight path angles and the velocity:

$$\psi = \tan^{-1}\left(\frac{v_y}{v_x}\right)$$
$$v_{xy} = \sqrt{v_x^{\,2} + v_y^{\,2}} \qquad \text{(Eqn 3-4)}$$
$$\gamma = \tan^{-1}\left(\frac{v_z}{v_{xy}}\right)$$

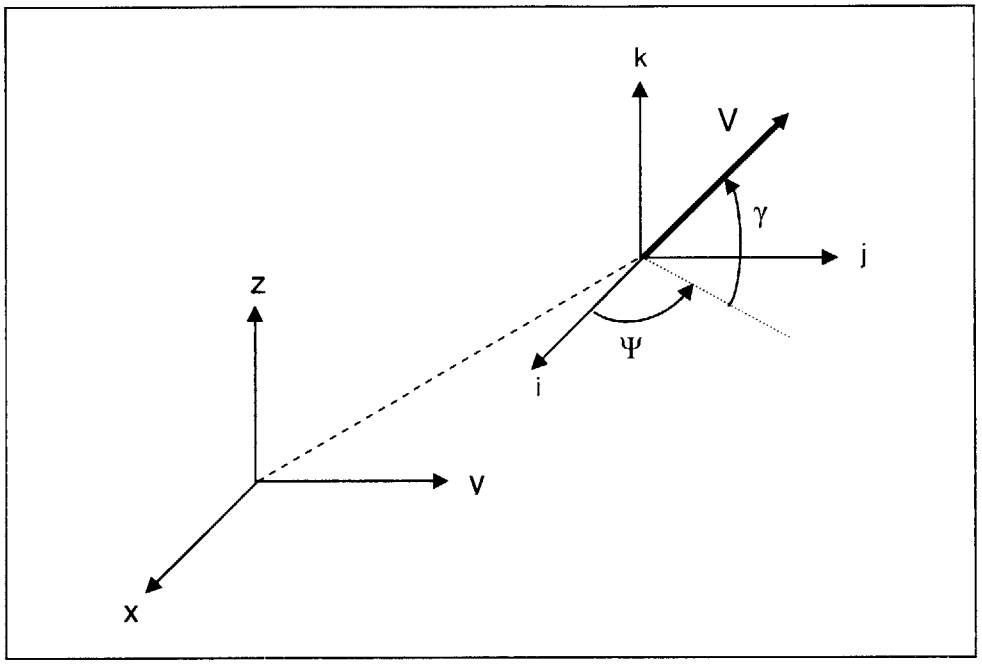

Figure 3-6 Velocity description for chromosome representation

This description of vehicle velocity orientation leads to the convenient instruction list chromosome representation, as described in [35]. The instruction list formulation consists of a string of commanded changes in speed, heading angle, and flight path angle that take place over a finite amount of time. Let us define an instruction for transition from time t_k to time t_{k+1} as:

$$I_k[t_k \rightarrow t_{k+1}] = \{\Delta V[t_k \rightarrow t_{k+1}], \Delta\Psi[t_k \rightarrow t_{k+1}], \Delta\gamma[t_k \rightarrow t_{k+1}]\} \qquad \text{(Eqn 3-5)}$$

Assume for the time being (this will be discussed in Section 3.4) that each instruction in any given instruction list takes place over a fixed time interval defined as $\Delta t_k = (t_{k+1} - t_k)$. With this, the instruction list can be described in general form as:

$$\{I_1[\Delta t_1],\ I_2[\Delta t_2], I_3[\Delta t_3],\ \ldots.\ , I_m[\Delta t_m]\} \qquad \text{(Eqn 3-6)}$$

where m represents the number of instructions in the list. Table 3-1 gives an example of an instruction list of m = 5. If each command were to take place over a five second interval, then a trajectory 25 seconds in length would be produced. Note that the trajectory in the table ends up with the same speed, heading, and flight path angle as it initiated with.

Table 3-1 Example instruction list chromosome

Time (sec)	ΔV (kts)	$\Delta\Psi$ (deg)	$\Delta\gamma$ (deg)
0 - 5	+8	0	0
5 - 10	0	+10	0
10 - 15	-16	0	+2
15 - 20	0	-15	+6
20 - 25	+8	+5	-8

The GA representation for TERA can thus be described as follows; each chromosome is simply a string of instructed state transitions over a discrete time interval, with each gene consisting of three values (not-binary) relaying the desired changes in speed, heading angle, and flight path angle. By varying the number of instructions in the list and the Δt_k for the instruction transitions within the list, the total time length of the trajectory as well as the dynamic resolution can be dictated.

While the chromosome representation described is straightforward and will prove to be convenient for imposing dynamic constraints, it is not a useful representation for evaluating the fitness of each population member. Although the exact fitness functions have yet to be described, we know intuitively that at least LOS and range to threats will affect survivability. Because of this, the fitness evaluation will need to take into account the location of the vehicle with respect to the threats within the inertial frame. The chromosome representation as shown in Table 3-1 obviously does not contain the inertial location of the vehicle, so a mapping between input and output space is required for fitness evaluation within the GA.

If we make the simplifying assumption that the changes in speed take place at a constant acceleration over the transition interval, and that changes in heading and flight path angle take place at constant turn rates, i.e.:

$$\dot{V}(\Delta t_k) = \frac{\Delta V(\Delta t_k)}{\Delta t_k}$$
$$\dot{\Psi}(\Delta t_k) = \frac{\Delta \Psi(\Delta t_k)}{\Delta t_k} \qquad \text{(Eqn 3-7)}$$
$$\dot{\gamma}(\Delta t_k) = \frac{\Delta \gamma(\Delta t_k)}{\Delta t_k}$$

then a very convenient analytical mapping can be derived. Begin with the speed of the vehicle, which can be described in the inertial coordinate frame as:

$$\vec{V} = \begin{bmatrix} V(t)\cos(\gamma)\cos(\Psi) \\ V(t)\cos(\gamma)\sin(\Psi) \\ V(t)\sin(\gamma) \end{bmatrix} \qquad \text{(Eqn 3-8)}$$

Since the acceleration and turn rates are assumed constant, the speed and angles can be expressed at any time $t > t_o$ as:

$$V(t) = V(t_o) + \dot{V}(t - t_o)$$
$$\Psi(t) = \Psi(t_o) + \dot{\Psi}(t - t_o) \qquad \text{(Eqn 3-9)}$$
$$\gamma(t) = \gamma(t_0) + \dot{\gamma}(t - t_o)$$

Substituting Equation 3-9 into Equation 3-8 and then integrating over time using known boundary conditions $V(t_o)=V_o$, $\Psi(t_o)= \Psi_o$, $\gamma(t_o)= \gamma_o$, $x(t_o)=x_o$, $y(t_o)=y_o$, and $z(t_o)=z_o$ yields equations for x(t), y(t), and z(t). These can be simplified without loss of generality (since we will only care about deltas in x, y, and z) by setting $t_o = 0$ to give:

$$x(t) = x_o +$$
$$\frac{1}{2(\dot{\gamma} - \dot{\Psi})^2}\left\{\begin{array}{l}(V_o + \dot{V}\cdot t)(\dot{\gamma} - \dot{\Psi})\sin\left[(\dot{\gamma} - \dot{\Psi})\cdot t + \gamma_o - \Psi_o\right] + \\ \dot{V}\cos\left[(\dot{\gamma} - \dot{\Psi})\cdot t + \gamma_o - \Psi_o\right] - \dot{V}\cos(\gamma_o - \Psi_o) - \\ V_o(\dot{\gamma} - \dot{\Psi})\sin(\gamma_o - \Psi_o)\end{array}\right\} + \qquad \text{(Eqn 3-10)}$$
$$\frac{1}{2(\dot{\gamma} + \dot{\Psi})^2}\left\{\begin{array}{l}(V_o + \dot{V}\cdot t)(\dot{\gamma} + \dot{\Psi})\sin\left[(\dot{\gamma} + \dot{\Psi})\cdot t + \gamma_o + \Psi_o\right] + \\ \dot{V}\cos\left[(\dot{\gamma} + \dot{\Psi})\cdot t + \gamma_o + \Psi_o\right] - \dot{V}\cos(\gamma_o + \Psi_o) - \\ V_o(\dot{\gamma} + \dot{\Psi})\sin(\gamma_o + \Psi_o)\end{array}\right\}$$

$$y(t) = y_o +$$
$$\frac{1}{2(\dot{\gamma}-\dot{\Psi})^2}\begin{Bmatrix}(V_o+\dot{V}\cdot t)(\dot{\gamma}-\dot{\Psi})\cos[(\dot{\gamma}-\dot{\Psi})\cdot t+\gamma_o-\Psi_o]-\\ \dot{V}\sin[(\dot{\gamma}-\dot{\Psi})\cdot t+\gamma_o-\Psi_o]+\dot{V}\sin(\gamma_o-\Psi_o)-\\ V_o(\dot{\gamma}-\dot{\Psi})\cos(\gamma_o-\Psi_o)\end{Bmatrix}+ \qquad \text{(Eqn 3-11)}$$
$$\frac{1}{2(\dot{\gamma}+\dot{\Psi})^2}\begin{Bmatrix}-(V_o+\dot{V}\cdot t)(\dot{\gamma}+\dot{\Psi})\cos[(\dot{\gamma}+\dot{\Psi})\cdot t+\gamma_o+\Psi_o]+\\ \dot{V}\sin[(\dot{\gamma}+\dot{\Psi})\cdot t+\gamma_o+\Psi_o]-\dot{V}\sin(\gamma_o+\Psi_o)+\\ V_o(\dot{\gamma}+\dot{\Psi})\cos(\gamma_o+\Psi_o)\end{Bmatrix}$$

$$z(t) = z_o - \frac{V_o}{\dot{\gamma}}[\cos(\gamma_o+\dot{\gamma}\cdot t)-\cos(\gamma_o)]+ \qquad \text{(Eqn 3-12)}$$
$$\frac{\dot{V}}{\dot{\gamma}^2}[\sin(\gamma_o+\dot{\gamma}\cdot t)-\sin(\gamma_o)]-\frac{\dot{V}\cdot t}{\dot{\gamma}}\cos(\gamma_o+\dot{\gamma}\cdot t)$$

Equations 3-10, 3-11, and 3-12 thus provide the means to map a given instruction list to a trajectory. Figure 3-7 illustrates the use of these equations to decode the instruction list found in Table 3-1. The O's represent the trajectory points found at each five second increment, while the solid line demonstrates how a higher resolution trajectory (in this case 20 Hz) can be found using the same equations.

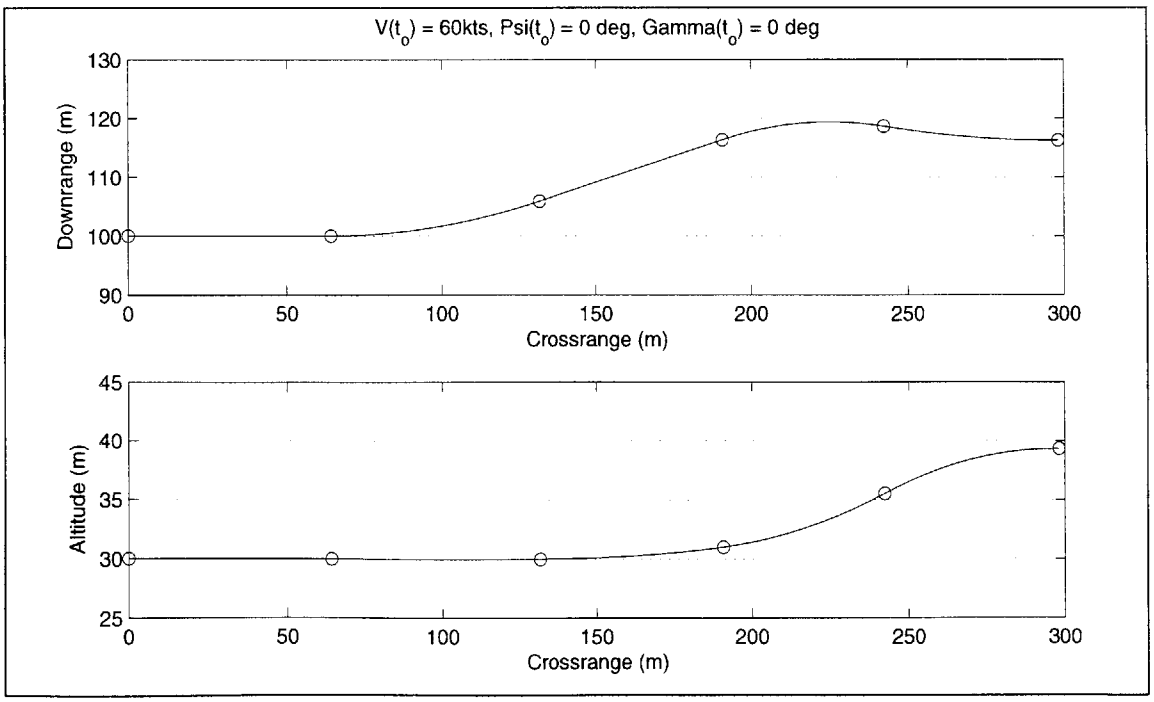

Figure 3-7 Trajectory created by instruction list from Table 3-1

3.2.1.1 Dynamic constraints

The requirements for TERA call for the creation of dynamically realizable trajectories. This can be achieved by limiting the range of command values at any given time to those values physically realizable by the vehicle. The fundamental need throughout the population initialization, crossover, and mutation operations is to be able to determine the allowable ΔV^+, ΔV^-, $\Delta\gamma^+$, $\Delta\gamma^-$, $\Delta\Psi^+$, and $\Delta\Psi^-$ based on the vehicle's state $\underline{r}_i = [V_i\ \Psi_i\ \gamma_i]$ at the beginning of instruction i (here $\Delta\_^+$ refers to the max increase in state _ and $\Delta\_^-$ refers to the max decrease in _). By setting the limits on command magnitudes, the GA is free to take actions such as random instruction selection without the possibility of creating dynamically infeasible trajectories. It is important to note that even though the discussion in this thesis pertains to a rotorcraft, the form of the command limits is general enough to make the GA trajectory generation method applicable to fixed wing aircraft as well. The dynamic limits must simply be altered appropriately for the vehicle.

The first step in determining the feasible set of commands for a specific instruction is to select Δv_{xy}^+ and Δv_{xy}^-, with v_{xy} as defined in Equation 3-4 (this is because the maneuvering bounds will be given on forward acceleration as opposed to rate of change of overall speed). By approximating the acceleration during a given instruction as constant, the required limits can be expressed as:

$$\Delta v_{xy}^{+} = \dot{v}_{xy}^{+} \cdot \Delta t$$
$$\Delta v_{xy}^{-} = \dot{v}_{xy}^{-} \cdot \Delta t \qquad \text{(Eqn 3-13)}$$

where $\dot{v}_{xy}^{+}$ represents the maximum forward acceleration, and $\dot{v}_{xy}^{-}$ represents the maximum forward deceleration. Figure 3-8 illustrates an example acceleration profile for a helicopter. The deceleration profile was modeled as a first order system response in velocity due to the lack of actual helicopter data. The first order time response is given by:

$$v_{xy}^{-}(t) = e^{-t/\tau} \cdot v_{xy}(0) \qquad \text{(Eqn 3-14)}$$

Differentiating with respect to time then yields the deceleration as:

$$\dot{v}_{xy}^{-} = -\frac{1}{\tau} e^{-t/\tau} \cdot \dot{v}_{xy}(0) \qquad \text{(Eqn 3-15)}$$

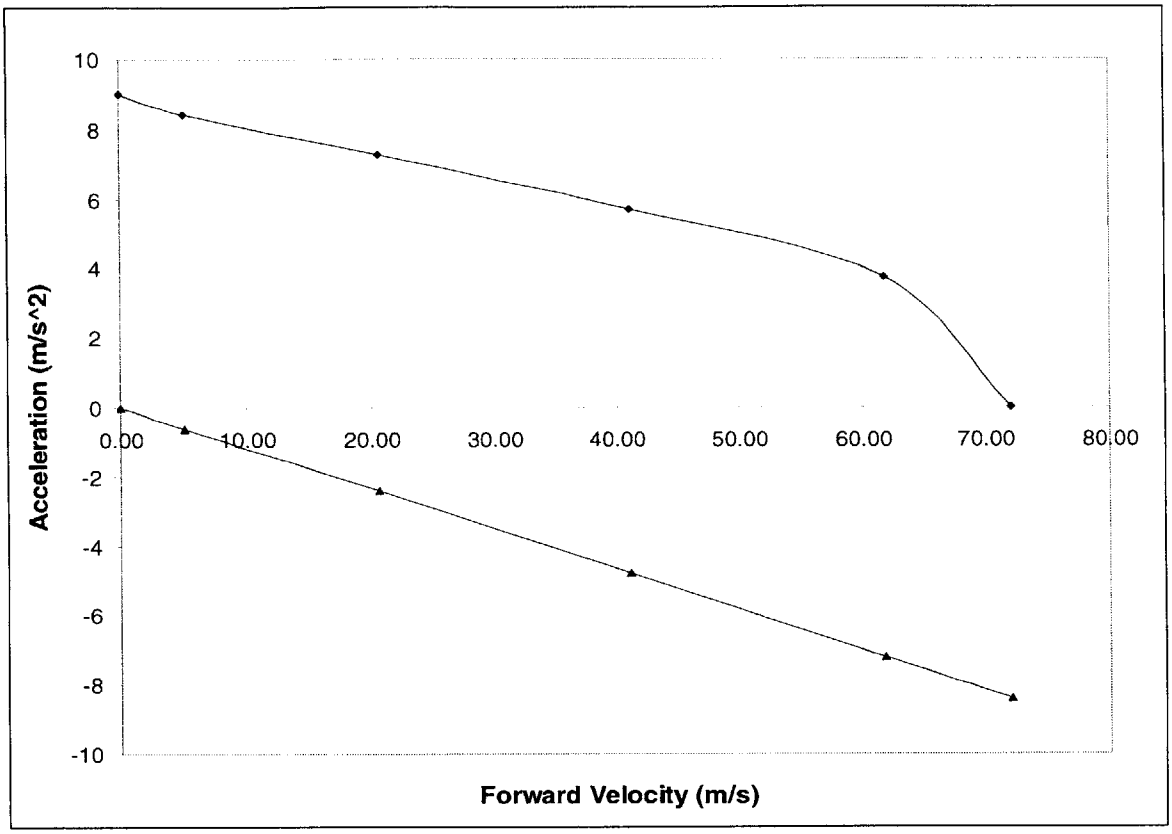

Figure 3-8 Acceleration profile for simple rotorcraft

The values used to define the bounds in Equation 3-13 are thus found from the acceleration limits shown in Figure 3-8. The overall change in speed once the value of Δv_{xy} has been chosen is given by:

$$\Delta V = \frac{v_{xy} + \Delta v_{xy}}{\cos(\gamma_o + \Delta\gamma)} \qquad \text{(Eqn 3-16)}$$

where $\Delta\gamma$ is selected from within the bounds on feasible flight path angle changes, which will be discussed next.

The $\Delta\Psi$ and $\Delta\gamma$ commands are coupled through a 2.5G maximum load factor on the vehicle, therefore the feasible range of values cannot be determined independently for Ψ and γ. We thus assume that during gene selection the values of $\Delta\gamma^{+}$ and $\Delta\gamma^{-}$ are determined, the $\Delta\gamma$ command selected, and the values of $\Delta\Psi^{+}$ and $\Delta\Psi^{-}$ then selected. The

value of $\Delta\gamma^-$ can be determined by approximating the downward acceleration the vehicle is capable of to 1G (this turns out to be a reasonable assumption for helicopters). By allowing the vehicle to pull -1G vertically over the time interval Δt the flight path angle is reoriented by a maximum of:

$$\Delta\gamma^- = \tan^{-1}\left(\frac{v_{z_o} - g\cdot\Delta t}{v_{xy_o} + \Delta v_{xy}}\right) - \gamma_o \qquad \text{(Eqn 3-17)}$$

where Δv_{xy} is the value selected from the range previously computed. For forward velocities other than zero there would be an absolute maintainable limit on the negative flight path angle, however due to the low altitude aspect of the threat avoidance problem no limit is imposed. When flying at very low altitudes the trajectory would become infeasible from ground strike before the terminal downward velocity is reached.

While the limit of $\Delta\gamma^-$ is determined by the downward acceleration the vehicle is capable of, the bottleneck which limits $\Delta\gamma^+$ is due to the relatively shallow rate of climb (ROC) capability. Figure 3-9 shows the ROC limit used, which is a function of speed. Using this information the value of $\Delta\gamma^+$ is given by the relation:

$$\Delta\gamma^+ = \sin^{-1}\left(\frac{ROC_{\max}(V_o)}{v_{xy_o} + \Delta v_{xy}}\right) - \gamma_o \qquad \text{(Eqn 3-18)}$$

Given the value of $\Delta\gamma$ selected from within the feasible range as described above, the range of feasible heading angle changes can be determined by noting the vehicle maximum turn loading of 2.5G's. Subtracting the centripetal loading due to the commanded $\Delta\gamma$ gives the remaining loading available for heading changes. The limits on $\Delta\Psi$ are thus:

$$\dot{\Psi}_{\max} = \frac{\sqrt{(2.5\cdot g)^2 - \left(V_o\cdot\frac{\Delta\gamma}{\Delta t}\right)^2}}{V_o}$$

$$\Delta\Psi^+ = \dot{\Psi}_{\max}\cdot\Delta t \qquad \text{(Eqn 3-19)}$$

$$\Delta\Psi^- = -\Delta\Psi^+$$

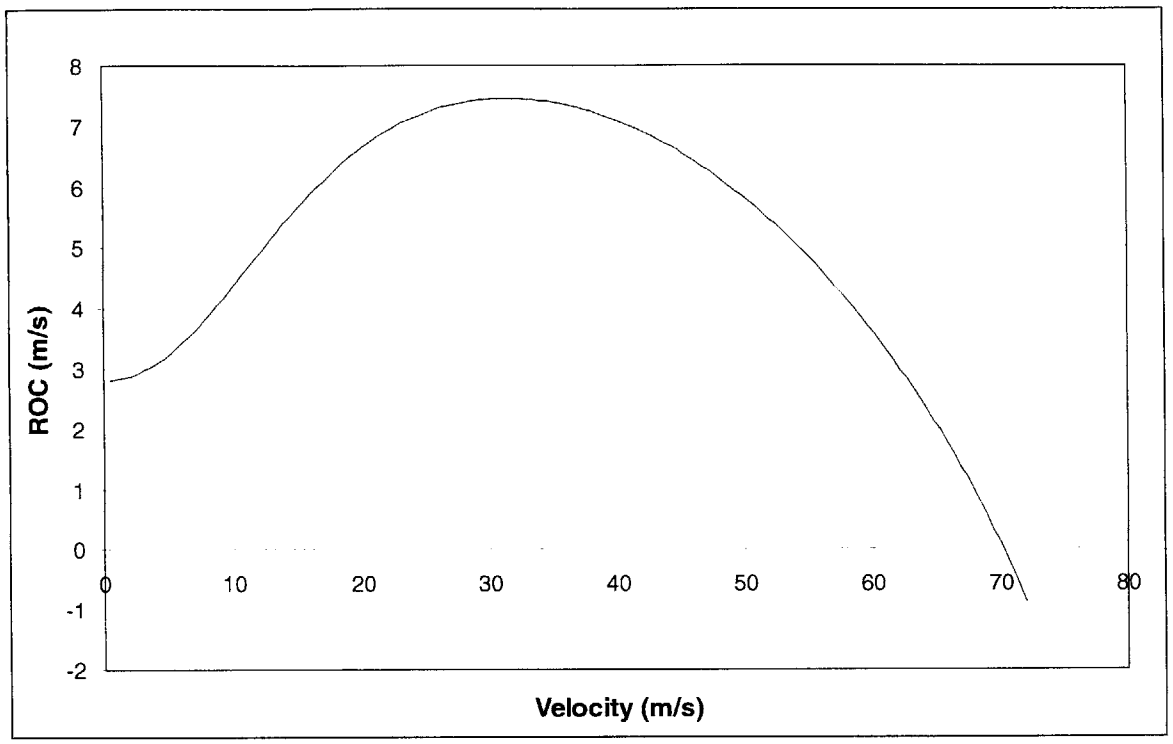

Figure 3-9 Rate of climb capability for rotorcraft

Once the feasible range on changes in heading angle has been determined, the value of $\Delta\Psi$ can be selected, yielding the feasible gene values:

$$I_i(\Delta t_i) = \{\Delta V, \Delta\Psi, \Delta\gamma\} \qquad \text{(Eqn 3-20)}$$

3.2.2 Population initialization

In order to begin the generational refinement process within the GA, an initial population must be provided. If each individual in the initial population is feasible and the crossover and mutation actions are forced to respect dynamic limits, then all trajectories generated through successive generations will remain dynamically feasible. We will thus require the population initialization to provide trajectories respecting maneuvering limits (ground constraints will be discussed in a moment).

The population initialization is achieved by randomly seeding each chromosome. Since the dynamic constraints can be expressed as an allowable range of ΔV, $\Delta\Psi$, and $\Delta\gamma$ based on the current V and γ, feasibility can be maintained in a given chromosome by simply starting with the first gene and seeding toward the last gene while keeping track of the magnitude of V and γ. The values of ΔV, $\Delta\Psi$, and $\Delta\gamma$ for a given gene are randomly

selected from a uniform distribution spanning the feasible range of values for the specific command, determined by the velocity and flight path angle at the initiation of the instruction being seeded. This consecutive random gene seeding process is performed for the desired length of each chromosome, and for the number of chromosomes desired for the population.

In addition to dynamic feasibility, care must be taken to reduce the number of ground strikes within the initial population. As will be discussed in Section 3.3, ground collisions are treated using soft constraints, i.e., trajectories that hit the ground will not be immediately pruned, but rather penalized within the fitness evaluation. Care must therefore be taken to prevent many or all of the initial population members from colliding with the terrain. Too many collision-infeasible trajectories would adversely affect the GA convergence by requiring the algorithm to spend much of its effort in eliminating ground strikes rather than reducing threat exposure. Ground strikes are reduced in the initialization by defining a maximum and minimum preferred altitude of operation. Each time a gene is seeded in a chromosome, the resulting inertial position is found using Equations 3-10, 3-11, and 3-12. If the trajectory is found to exceed one of the limits, the subsequent $\Delta\gamma$ is limited to the maximum or minimum feasible value depending on if the trajectory breaks the acceptable floor or ceiling altitudes, respectively. The minimum altitude can be set to ground level or some other small value to force the trajectories to pull up prior to hitting the ground. This concept is illustrated in Figure 3-10. Instruction I_n causes the trajectory to go below the minimum altitude, causing the value of $\Delta\gamma^+$ to be selected for gene I_{n+1}. Likewise, $\Delta\gamma^-$ would be selected if the maximum altitude was broken.

3.2.3 Crossover

Much of the power of the GA as a global directed random search agent comes from the crossover operation [34]. However, if not implemented intelligently based on the problem to be solved, crossover can lose much, if not all of its intended utility. Consider the waypoint chromosome representation discussed in Section 3.1.1. Each chromosome is comprised of an ordered list of waypoints to follow. It is assumed that the vehicle travels any given path in straight line segments connecting each waypoint. With

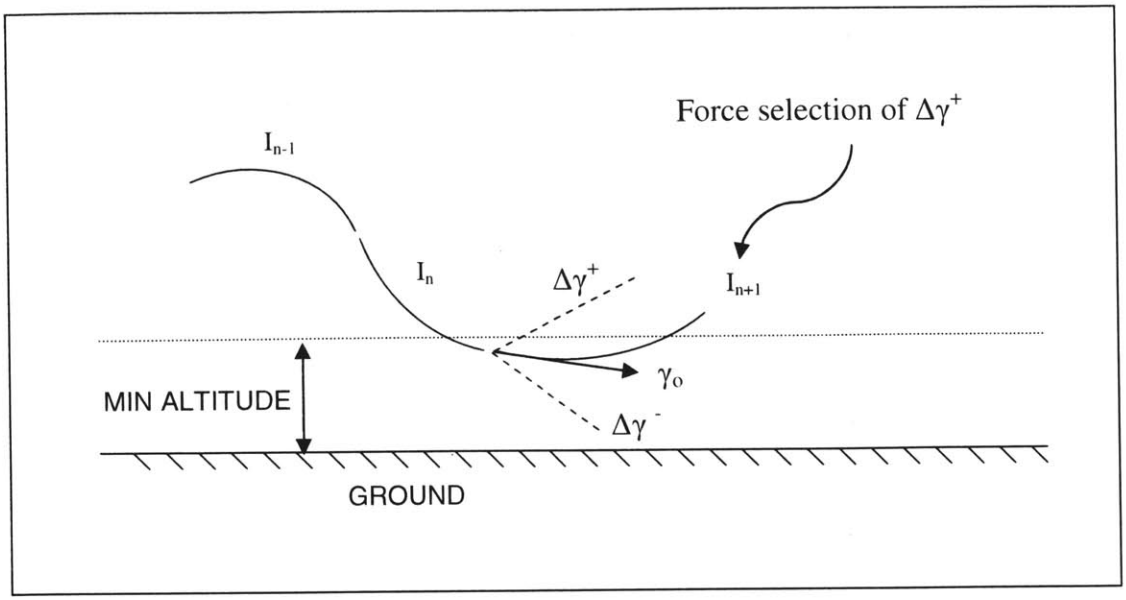

Figure 3-10 Heuristic for minimizing ground collisions in initialization

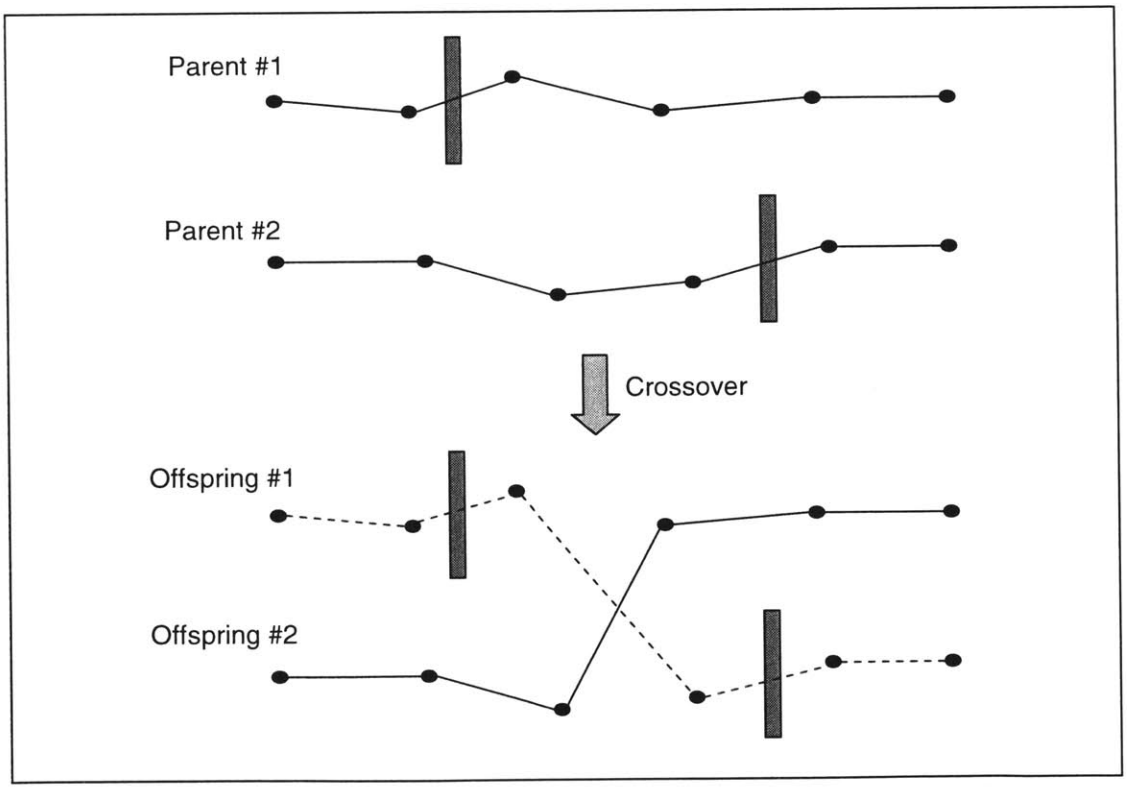

Figure 3-11 Crossover result for waypoint formulation

this representation a single point crossover has the effect illustrated in Figure 3-11. This crossover procedure was used in [30] to successfully navigate very complicated mazes. In this case the crossover operation preserves the inertial position of the portions of each

chromosome that are recombined in the resulting offspring. Figure 3-11 makes it clear that two infeasible chromosomes can be combined through crossover to create a feasible offspring.

Now consider the result of a simple single point crossover using the instruction list formulation. Each gene in the chromosome gives a *relative* change in V, Ψ, or γ applied over the time span of the gene. When two chromosomes are simply split and re-combined the effect is to shift the trajectory created by the latter portion of the second parent chromosome to line up with the end of the initial portion of the first parent (see Figure 3-12). Depending on the nature of the problem to be solved, this may or may not be the desired result. For our purposes fitness is a function of inertial location along a trajectory, as range and LOS affect risk. If a piece of a well performing trajectory were to be shifted to a new location it would no longer retain its desirable characteristics, and thus turn the crossover operation into a random mechanism, as opposed to a directed one.

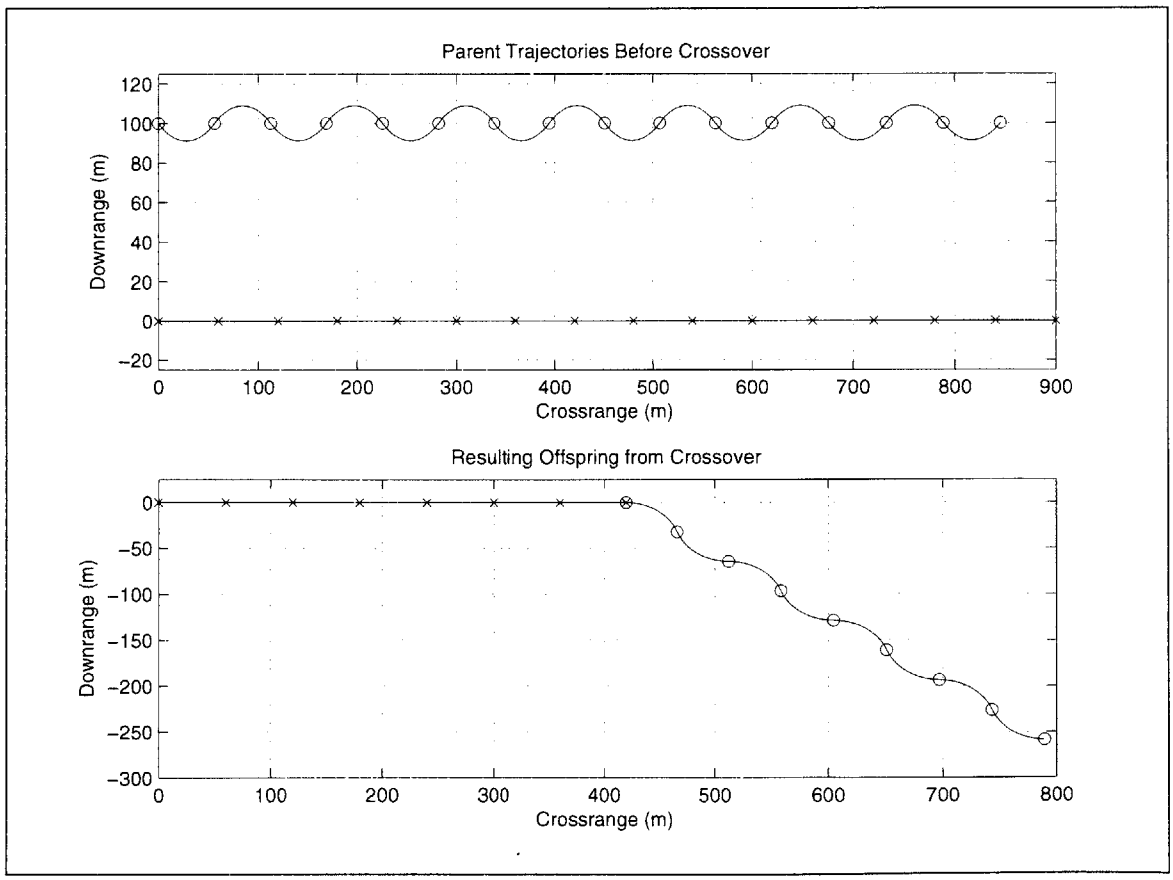

Figure 3-12 Crossover result for instruction list formulation

The crossover method used for TERA retains the inertial location of the original parent trajectories (as is the case in Figure 3-11) in order to preserve positive trajectory traits. In the waypoint formulation this was achieved by connecting the two chromosome segments with a straight line and calling the job done; however, the instruction list formulation makes the problem more difficult. Since TERA is required to maintain dynamic feasibility while transitioning from one trajectory segment to another, a process which extends the work presented in [33] to include altitude was developed to generate the additional transition instructions.

3.2.3.1 Crossover patching

The general problem that must be solved during the crossover patching operation is to join the end of one segment of trajectory (S1) with the beginning of a different trajectory segment (S2). This requires the generation of a set of genes to transition the vehicle from a given initial state $\underline{x}_1$ to a final state $\underline{x}_2$ defined by the trajectory segments to join, where $\underline{x} \equiv [x\ y\ z\ V\ \Psi\ \gamma]$. An example chromosome patching problem is shown in Figure 3-13. Both trajectories are restricted to the x-y plane for ease of illustration, however fully 3D trajectories will be discussed as this section progresses.

The crossover patching operation proceeds in two phases. The first phase consists of a series of acceleration or deceleration instructions which are added to the end of S1 in order to match up the velocity at the end of the new S1 with S2. The second phase consists of the generation of constant velocity turning instructions for joining the remaining five states. The velocity matching phase appends one gene at a time to the end of S1. Each gene commands the maximum acceleration (or deceleration) until the speed at the end of the last gene added equals the initial speed of S2. If the difference in speed can be achieved in one Δt, then only one gene will be added during the first phase. In addition, Ψ and γ are commanded to point the velocity vector in the direction of the initial (x, y, z) location of S2. Figure 3-14 shows the resulting phase one instructions for the example problem presented in Figure 3-13 (velocity matching instructions are labeled with diamonds).

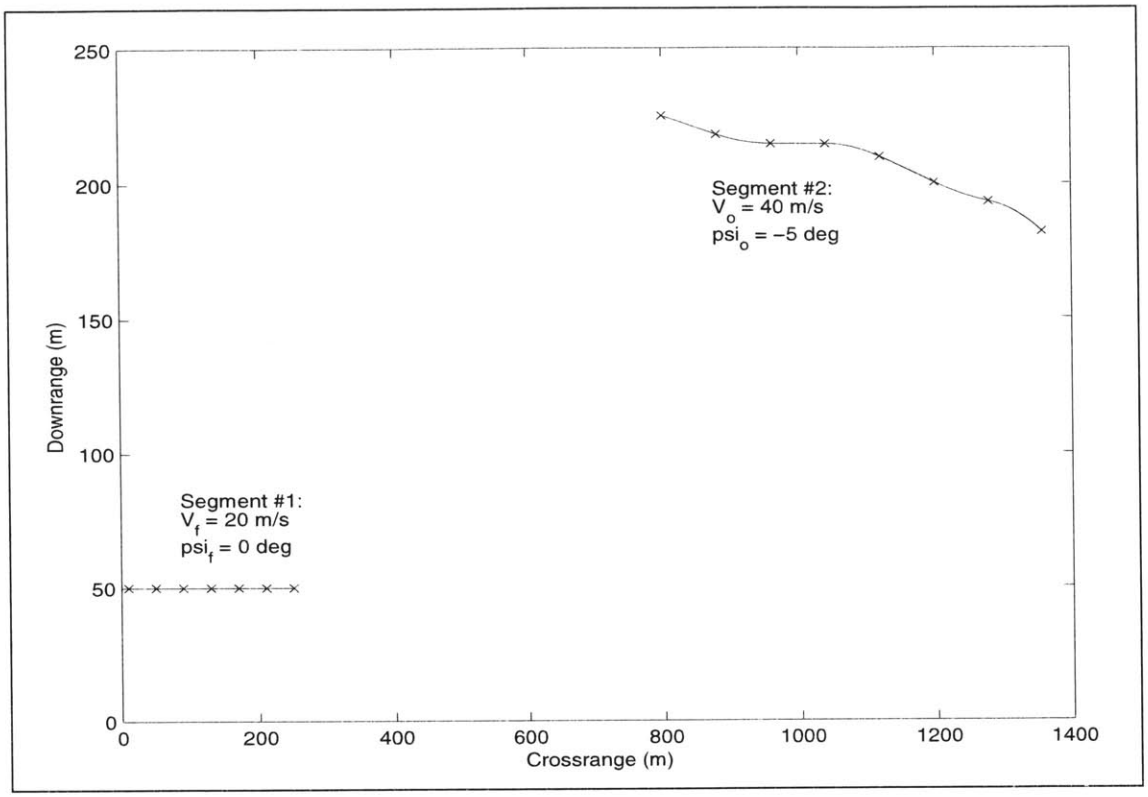

Figure 3-13 Initial crossover patching problem

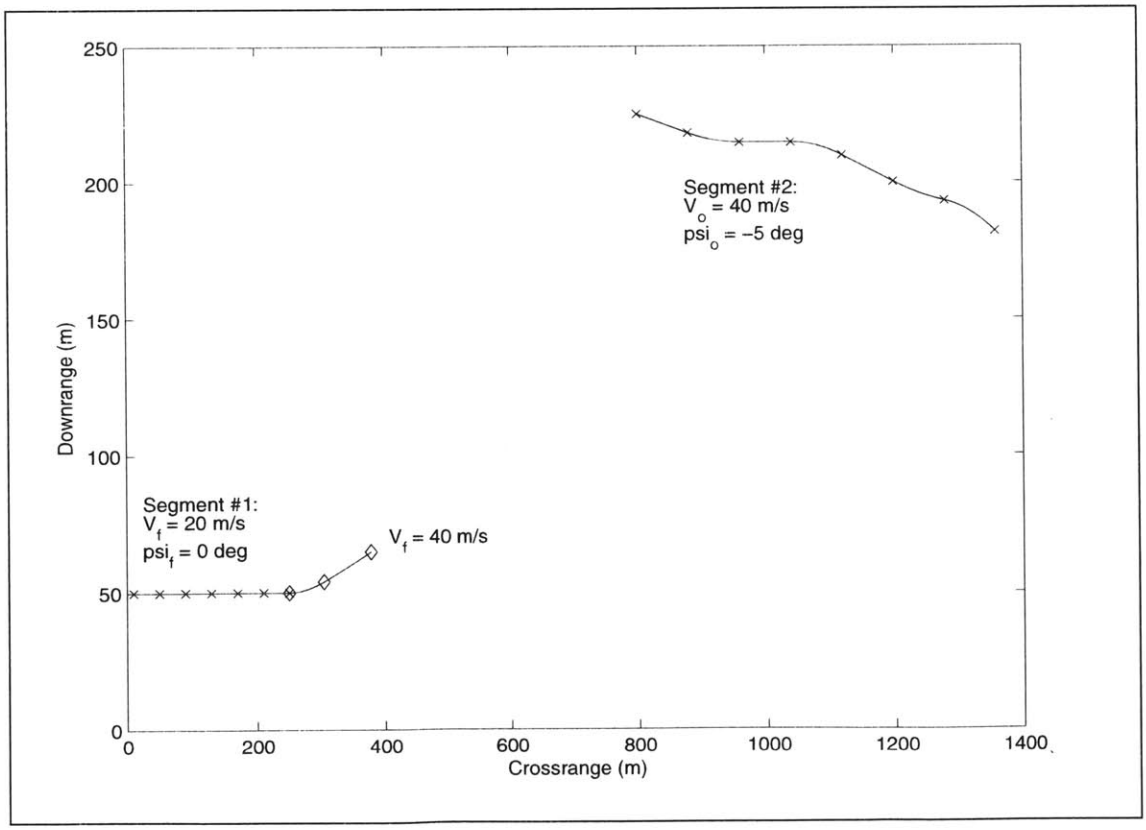

Figure 3-14 Result of velocity matching phase in crossover patching

The second phase requires the use of one or both of two different geometric point joining methods, first proposed in [33]. Both methods form a 2D solution joining two points with corresponding heading angles with circular segments. The first method, which will be referred to as the 'S' curve method (SCM), is shown in Figure 3-15. Two circles of equal radius R are formed such that each point (P_1 and P_2 in figure) is on the circumference of one of the circles, and the heading angle of each point is tangent to the adjoining circle. The radius is defined such that the circles touch at exactly one point, labeled 'C' in the figure.

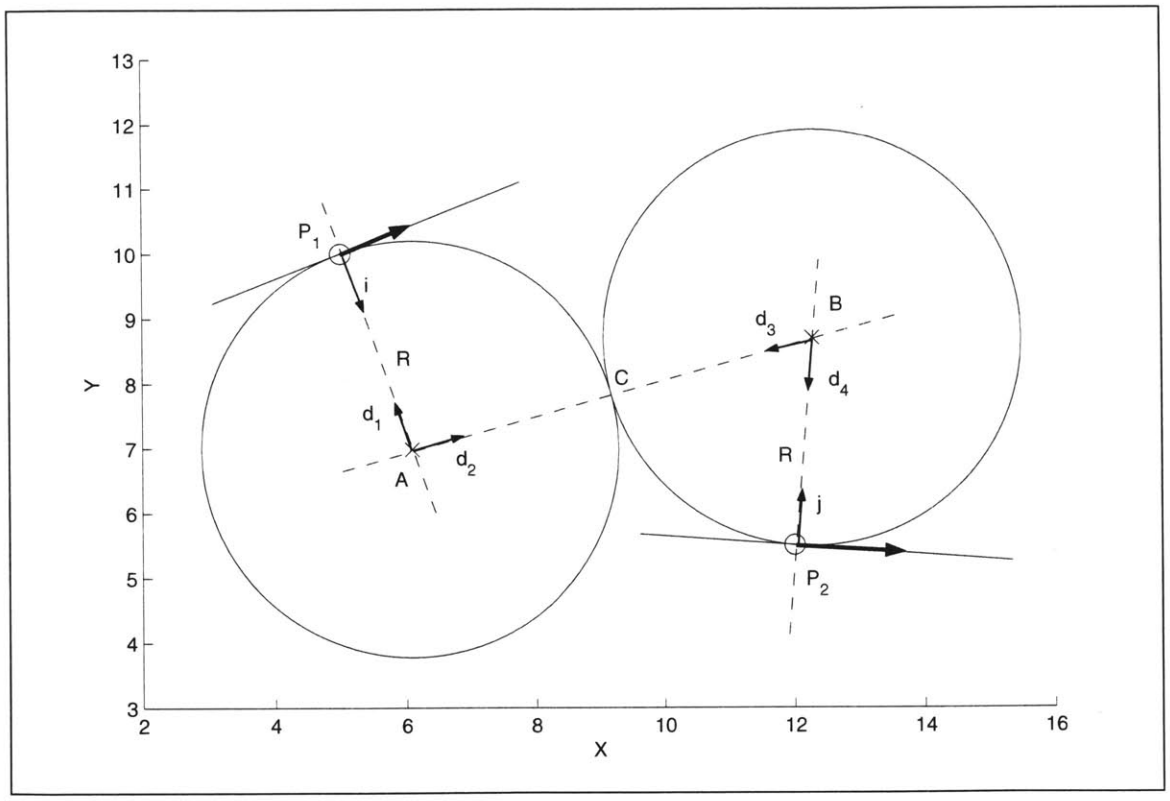

Figure 3-15 'S' curve method for 2D point joining

From the geometry we can define five scalar equations with five unknowns:

$$\underline{A} = \underline{P}_1 + R \cdot \underline{i}$$
$$\underline{B} = \underline{P}_2 + R \cdot \underline{j} \qquad \text{(Eqn 3-21)}$$
$$R = \frac{\|\underline{A} - \underline{B}\|}{2}$$

where A, B, and unit vectors i and j are defined as illustrated in Figure 3-15. Solving these equations leads to a quadratic which yields the value of R:

$$\begin{aligned} 0 = &(-4 + i^T i - 2 \cdot i^T i + j^T j) \cdot R^2 + \\ &(2 \cdot P_1^T i - 2 \cdot P_1^T j - 2 \cdot i^T P_2 + 2 \cdot P_2^T j) \cdot R + \\ &(P_1^T P_1 - 2 \cdot P_1^T P_2 + P_2^T P_2) \end{aligned} \qquad \text{(Eqn 3-22)}$$

The second geometric solution method, which will be referred to as the circle-in-circle method (CCM), is shown in Figure 3-16. As in SCM, two circles are formed tangent to the points to be joined, however, in CCM one circle is contained within the other circle. The two circles touch at exactly one point, which is denoted as C in Figure 3-16. The circle construction begins by forming the lines t_1, t_2, and t_3 (Figure 3-16). The lines t_1 and t_2 have the heading angles corresponding to P_1 and P_2, respectively, and line t_3 intersects points P_1 and P_2. The line t_4 is formed such that it is parallel to t_3, and the distances $|Cq| = |P_1q|$ and $|Ch| = |P_2h|$, where q and h are defined as the intersection points of t_1 with t_4 and t_2 with t_4, respectively. Finally, the line t_5 is defined as the line passing through points C, N, and M, where N and M define the centers of the two circles as illustrated in Figure 3-16. Due to the geometry of the formulation t_5 is perpendicular to line t_3. This allows the definition of the unit vector j (as shown in the figure) which will be used in the determination of R_1 and R_2.

In this case, two sets of four equations in four unknowns can be defined by noting that the points N and M can each be described in multiple ways:

$$\begin{aligned} \underline{N} &= \underline{P}_1 + R_1 \cdot \underline{r} \\ \underline{N} &= \underline{W} + \eta \cdot \underline{j} \end{aligned} \qquad \text{(Eqn 3-23)}$$

$$\begin{aligned} \underline{M} &= \underline{P}_2 + R_2 \cdot \underline{s} \\ \underline{M} &= \underline{W} + \beta \cdot \underline{j} \end{aligned} \qquad \text{(Eqn 3-24)}$$

where the unit vectors r and s are unknown, and R_1, R_2, η, and β are unknown scaling factors. The following expression for R_1 can then be found by simultaneously solving the four equations in Equation 3-23, with the unknown scalar η being substituted out of the equation during the algebraic manipulations:

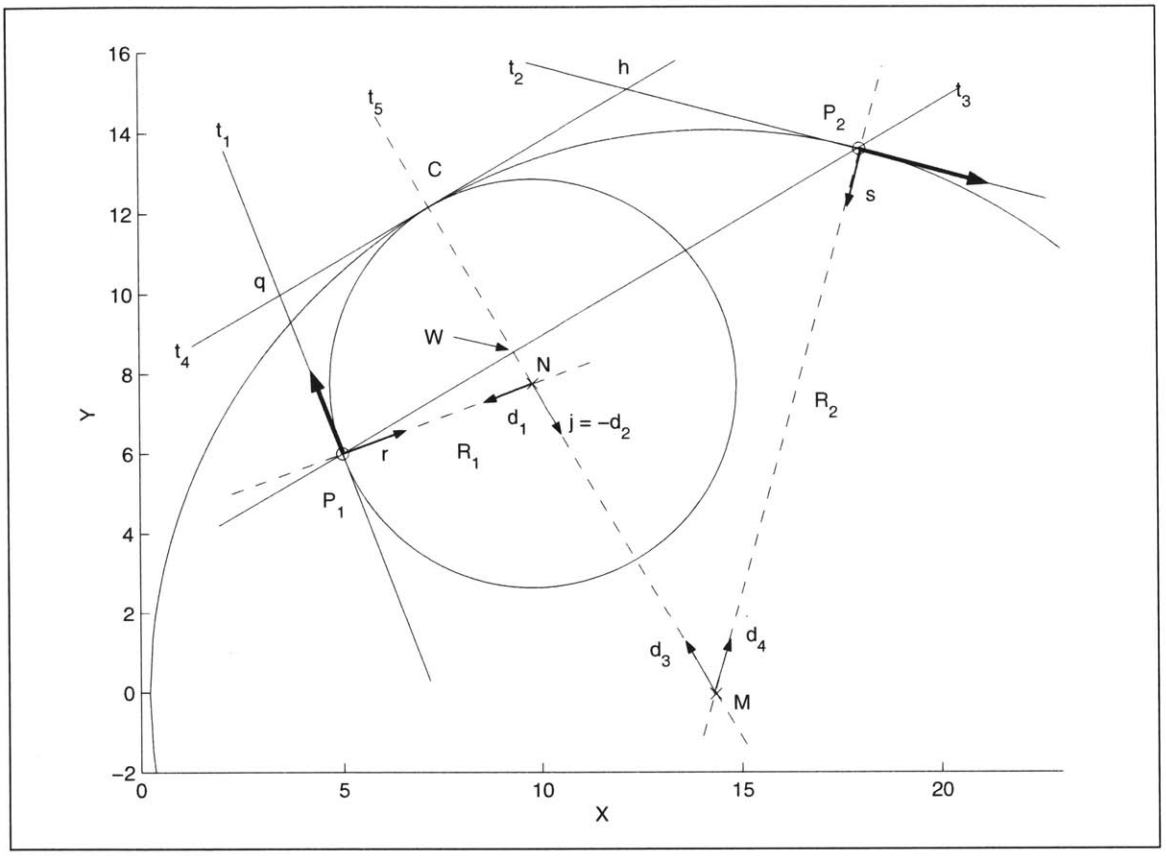

Figure 3-16 Circle-in-circle method for 2D point joining

$$R_1 = \frac{W_x j_y + P_{1y} j_x - W_y j_x - P_{1x} j_y}{r_x j_y - r_y j_x} \qquad \text{(Eqn 3-25)}$$

Likewise, solving Equation 3-24 for R_2 after substituting β out yields:

$$R_2 = \frac{(W_x - P_{2x}) j_y + (P_{2y} - W_y) j_x}{s_x j_y - s_y j_x} \qquad \text{(Eqn 3-26)}$$

In order to solve for R_1 and R_2 we must know the location of point W. The point W can be described in relation to the point P_1 using the unit vector pointing from P_1 to P_2, and the fact that W lies on the line connecting P_1 to P_2:

$$W = P_1 + |P_1 W| \frac{P_1 - P_2}{\sqrt{(P_1 - P_2)^T (P_1 - P_2)}} \qquad \text{(Eqn 3-27)}$$

where the length $|P_1 W|$ is given by the relation:

$$|P_1W| = \frac{|CW|}{\tan(\theta_1/2)} \qquad \text{(Eqn 3-28)}$$

and the length |CW| given by:

$$|CW| = \frac{\sqrt{(P_1 - P_2)^T (P_1 - P_2)}\tan(\theta_1/2)}{\left[1 + \frac{\tan(\theta_1/2)}{\tan(\theta_2/2)}\right]} \qquad \text{(Eqn 3-29)}$$

Here the θ_1 measures the angle between $\overline{P_1 q}$ and $\overline{P_1 W}$, and θ_2 measures the angle between $\overline{P_2 h}$ and $\overline{P_2 W}$. Equations 3-28 and 3-29 are derived from the geometry in Figure 3-16, specifically the fact that line t_4 is parallel to line t_3, and that t_5 is perpendicular to both of those.

The SCM and CCM form the heart of the joining phase of crossover patching. The chromosome joining proceeds in two steps. The first step begins with joining the final (x, y, Ψ) of the appended S1 with the initial (x, y, Ψ) of S2 through the use of the SCM or CCM. The method used is chosen based on whether the initial turning direction from S1 is the same as the turn direction approaching S2 (CCM), or whether the turning directions are opposite (SCM). The solution returned represents the path to follow if the transition were confined to the horizontal plane, and provides the ΔΨ instructions required for the transition. The radius of each circle determines the turning rate required while traveling along the arc segment defined by circle i via the relation:

$$\dot{\Psi}_i = V / R_i \qquad \text{(Eqn 3-30)}$$

where V is the constant speed during the transition.

The amount of time for which the constant turning rate is applied is found by dividing the length of the circular arc traveled by the speed. The time spent on arc one and arc two respectively are given in Equation 3-31, with the vectors d_1, d_2, d_3, and d_4 defined in Figure 3-15 for SCM, or Figure 3-16 for CCM. Since there are two arc segments to traverse there will be a minimum of two genes required for the chromosome joining, one with instruction $\Delta\Psi_1 = \dot{\Psi}_1 t_1$ and the other with $\Delta\Psi_2 = \dot{\Psi}_2 t_2$.

$$t_1 = \frac{\left[\cos^{-1}\left(\frac{d_1^T d_2}{\sqrt{d_1^T d_1}\sqrt{d_2^T d_2}}\right)R_1\right]}{V}$$

$$t_2 = \frac{\left[\cos^{-1}\left(\frac{d_3^T d_4}{\sqrt{d_3^T d_3}\sqrt{d_4^T d_4}}\right)R_2\right]}{V} \qquad \text{(Eqn 3-31)}$$

For this thesis, however, the chromosome patching is not necessarily performed with two genes. For the purposes of fitness evaluation, we wish to use a fixed Δt for each gene in each chromosome. Because of this, the number of genes required for traveling arc segment i is $t_i/\Delta t$, which typically is not an integer value. The process developed to determine the number of genes and the $\Delta\Psi$ instruction for each gene to most closely approximate the SCM or CCM join solution is the second step in the joining operation, and proceeds as follows:

1. Find the number of genes to use for the first arc segment, $n = t_1/\Delta t$. If $n < 1$ use one gene. If $n > 1$ find the number of genes, n', by rounding n to the closest integer.
2. Use the instruction $\Delta\Psi = \dot{\Psi}_1 \Delta t$ for the first n'-1 genes. The last gene uses instruction $\Delta\Psi = \Psi_C - \Psi_f^{n'-1}$, where Ψ_C is the heading angle at the transition from the first to second arc segment, and $\Psi_f^{n'-1}$ is the heading angle at the end of the n'-1 gene.
3. Find two instruction lists for arc segment two with number of genes m', found by rounding up $m = t_2/\Delta t$, and m'', found by rounding down m. The $\Delta\Psi$ instructions for each of these lists are found as in Step 2.
4. Decode the two instruction lists and select the set of instructions that minimizes the distance between the final (x, y) location using each list and the initial (x, y) location of S2.

The process outlined above is illustrated in Figure 3-17 for the case of joining the two trajectory segments in Figure 3-14. This particular instance uses the SCM method, however the process is identical when using the CCM. The joining circles found by SCM are shown, along with the trajectories that result. Note that the two trajectories very nearly coincide, however the trajectory denoted by the diamonds has one less segment than the other. The longer chromosome ends up closer to the goal, and therefore is selected.

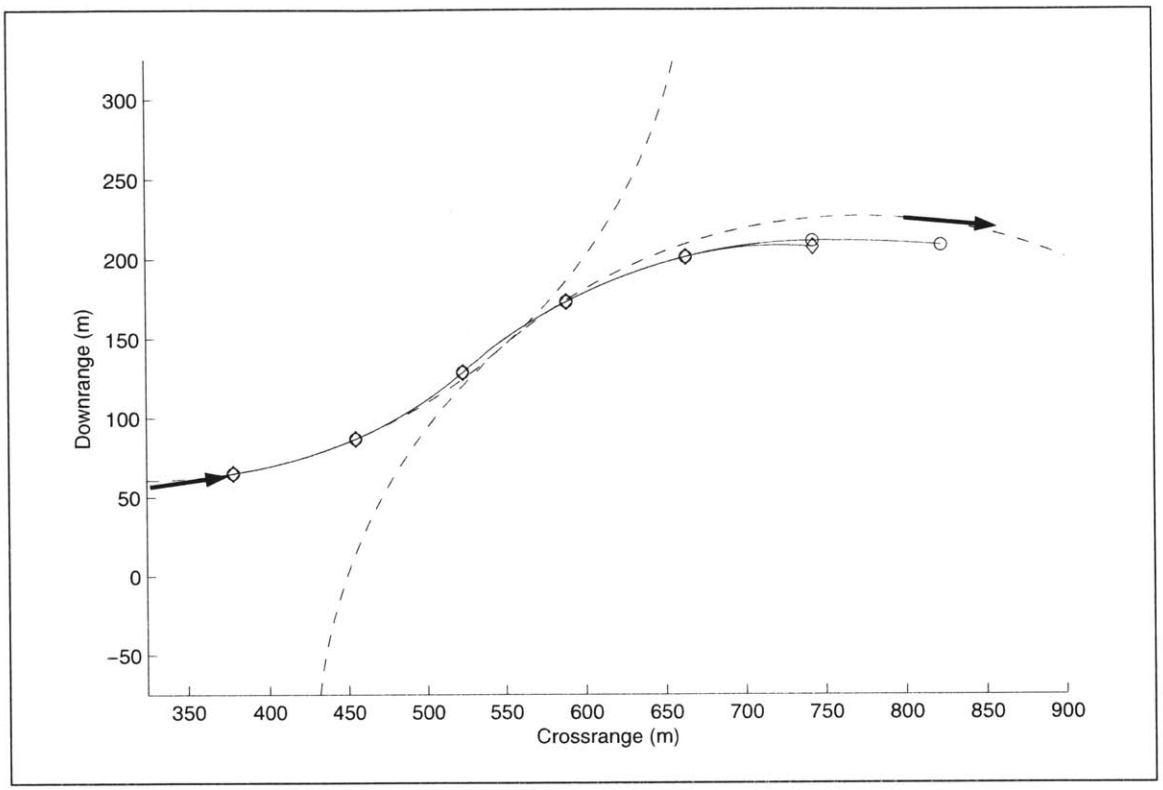

Figure 3-17 Gene creation in crossover

This example illustrates that the genes found approximate the SCM solution, however the final location of the patching trajectory is shifted from the desired final location. Because of this the segment S2 must be shifted to align with the joining trajectory. This is shown in Figure 3-18 which presents the final crossover trajectory for the problem depicted in Figure 3-13. While the crossover patch does not perfectly join S1 and S2, the shift required is relatively small on the scale of the trajectory and still retains the nature of each segment.

The example presented in Figure 3-18 is only a 2D problem, and we still require 3D functionality. The process followed for determining the $\Delta\gamma$ instructions is nearly identical to that used to find the $\Delta\Psi$ instructions. The SCM or CCM is called a second time to create a solution joining the altitude and flight path angles of the end of the appended S1 (z_1, γ_1) and the beginning of S2 (z_2, γ_2). To create the point joining solution, the values $P_1 = (0, z_1)$ with angle γ_1 and $P_2 = (\alpha*\Delta t, z_2)$ with angle γ_2 are used, where α is the total number of genes required for the instruction list generated in the x-y plane solution. An example 3D crossover patching is shown in Figure 3-19, with the transition

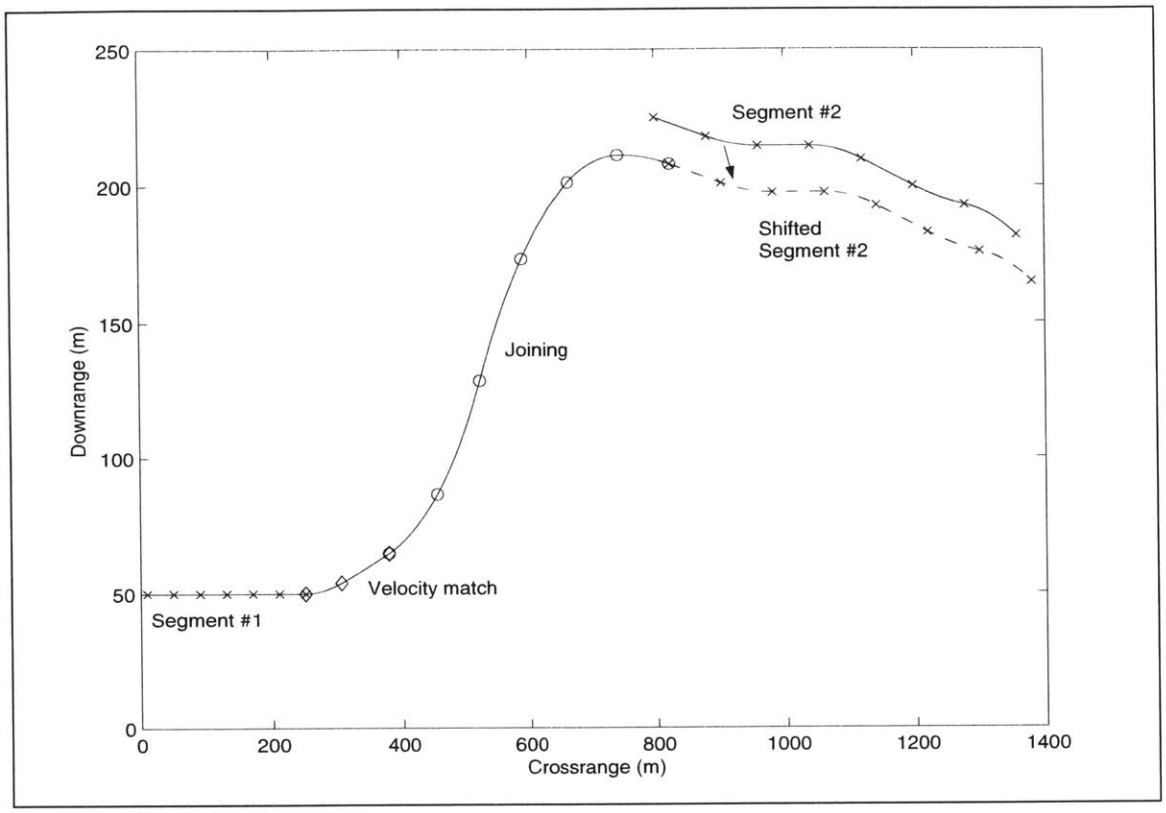

Figure 3-18 Final solution for crossover patching example

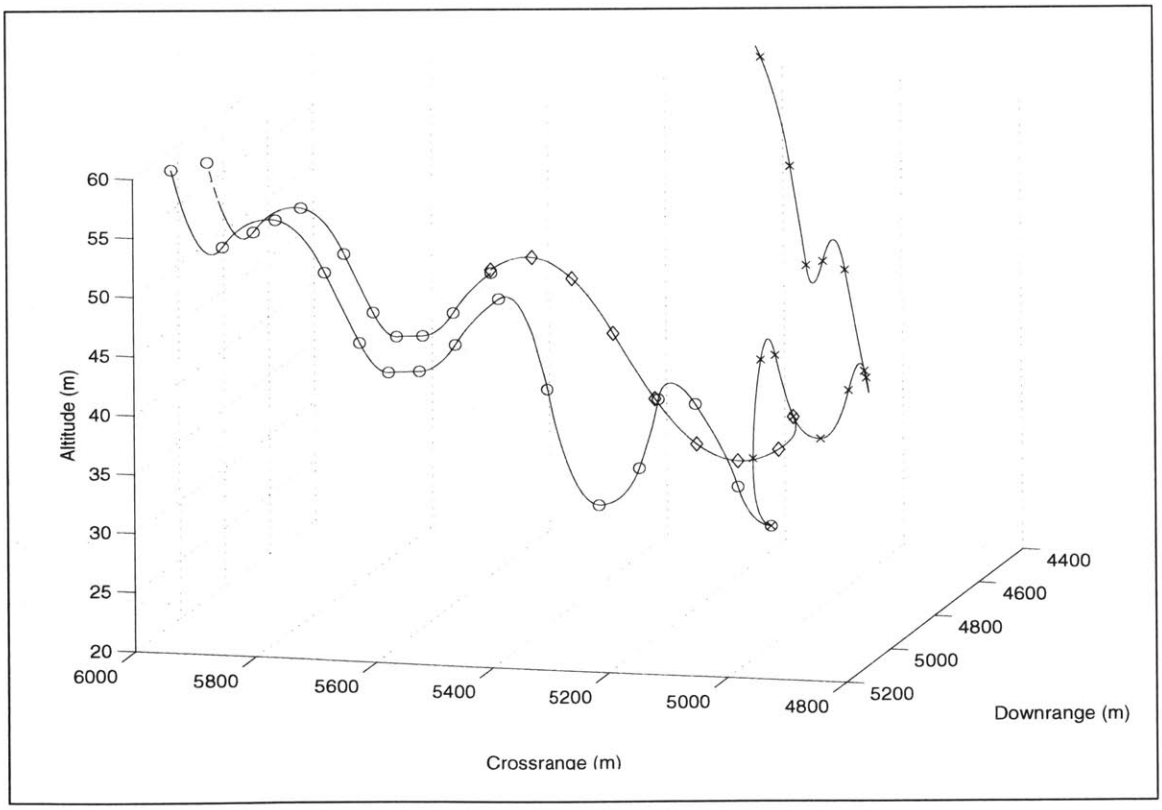

Figure 3-19 Three dimensional crossover example

trajectory segment denoted with diamonds. In this example the horizontal plane shift due to the patching is 37 m, while the altitude shift is 2.6 m.

The final step is to check for the dynamic feasibility of the crossover solution. Each gene in the joining segment of the chromosome is simply checked for feasibility against the dynamic limits as discussed in Section 3.2.1.1. If any $\Delta\Psi$ or $\Delta\gamma$ command is beyond the vehicle limits, the solution is deemed infeasible (dynamically) and the crossover patching operation returns failure. Crossovers that intersect the terrain are allowed to occur; however, these trajectories are penalized in the fitness evaluation (see Section 3.3).

3.2.4 Mutation

Each time a gene is selected for mutation (using the chosen probability of mutation) a random integer consisting of either 1, 2, or 3 is drawn, corresponding to the mutation of V, Ψ, or γ, respectively. A new value of the specific command to mutate is randomly selected from the feasible range of values based on the vehicle state at the beginning of the instruction and the values of the other two commands that are not being mutated. Figure 3-20 shows the results of two different chromosome mutations. The upper subplot shows the result of a mutation in Ψ on a constant altitude trajectory, while the lower subplot shows a mutation in γ on a constant heading trajectory ($\Psi = 0$). Note the two mutations shown are separate instances; the upper and lower trajectories are unrelated.

We would like the mutation operation to have the potential to dislodge a trajectory from a local minimum when any of the three commands in a given gene are selected for mutation. However, notice the resultant trajectory caused by the γ mutation in Figure 3-20. The mutation serves to shift the flight path angle at all downstream locations, which in this case makes the trajectory infeasible (assuming 0 meters altitude is the ground). Since the desired trajectory output from TERA should follow the terrain contour to take full advantage of terrain masking, it is highly unlikely that the dramatic shifts in altitude produced by the basic γ mutation would significantly reduce the cost of the original trajectory. Because of this the γ mutation is altered in an attempt to make it more productive.

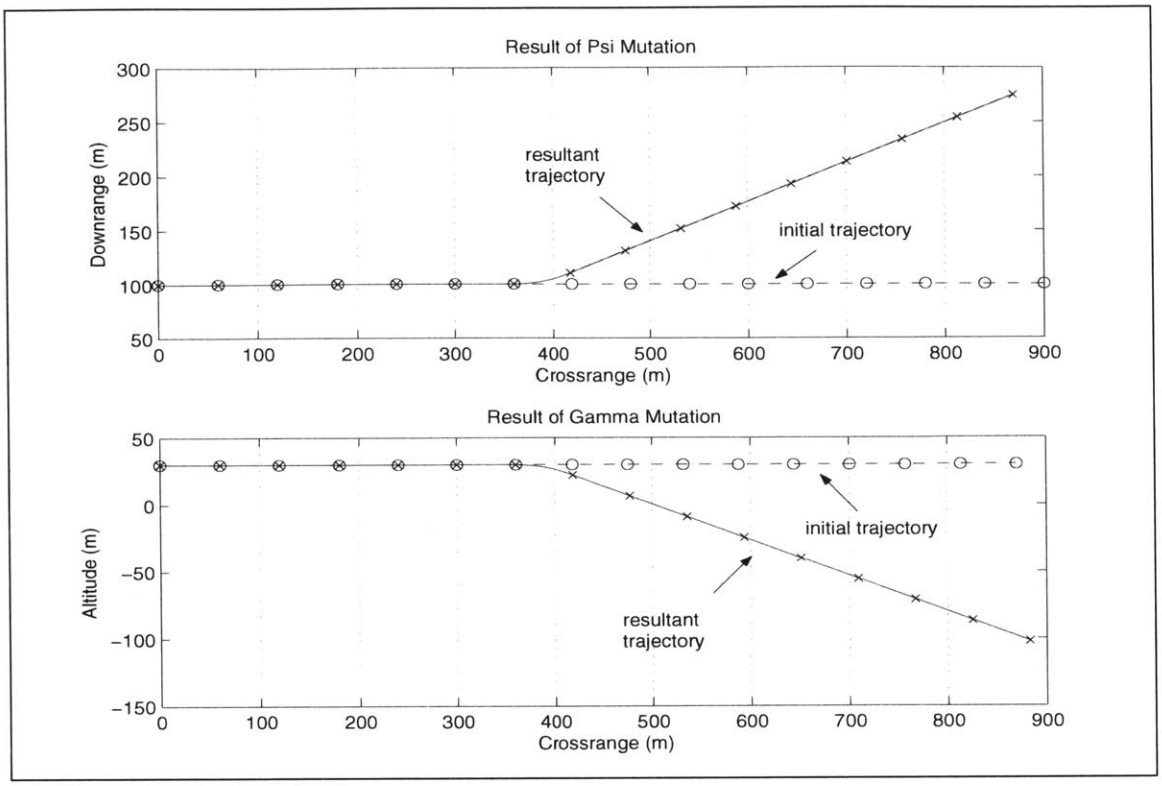

Figure 3-20 Results of random chromosome mutation

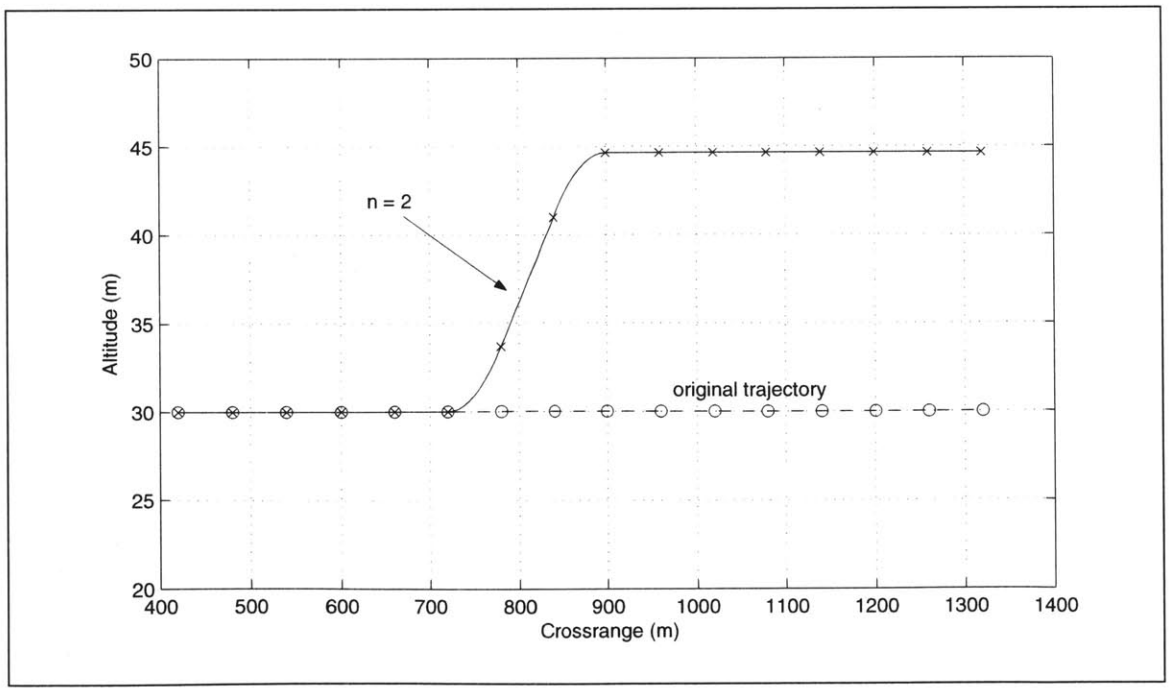

Figure 3-21 Modified γ mutation with altitude shift

Instead of mutating a given γ command and allowing it to propagate for the remainder of the trajectory, the $\Delta\gamma$ applied at a gene is subtracted from a subsequent gene. The number of genes the γ mutation is allowed to propagate is itself a random variable $n \in \{1,2,...,q\}$, where q is set by the user. If n=1 the $\Delta\gamma$ is subtracted out of the gene following the mutated gene. The effect of this mutation is a shift in altitude along the remaining trajectory, which is illustrated in Figure 3-21. Note that it may not be dynamically feasible to subtract out the required $\Delta\gamma$ in one gene. In this case the maximum $\Delta\gamma$ available for each gene is subtracted until the total $\Delta\gamma$ added in the mutation is removed.

3.2.4.1 Mutation repair

While the mutation operation will not pick an infeasible command to mutate a given gene, it does create the possibility of causing a gene "downstream" in the chromosome to become infeasible. For example, Table 3-2 contains a list of feasible commands for a given vehicle, as well as the same list with a mutation performed on the first instruction. The 4th column depicts the progression of speed along the trajectory, and the 8th column shows the same after the mutation. If the vehicle's max speed is 70 m/s, then the mutation, while in and of itself is feasible, has effectively made other genes infeasible within the same trajectory.

Table 3-2 Example of infeasibility caused by mutation

Original List				Mutated List			
ΔV (m/s)	ΔΨ (deg)	Δγ (deg)	V (m/s)	ΔV (m/s)	ΔΨ (deg)	Δγ (deg)	V (m/s)
0	0	0	55	6.2	0	0	61.2
5	0	0	60	5	0	0	66.2
5	0	0	65	5	0	0	71.2
5	0	0	70	5	0	0	76.2
0	0	0	70	0	0	0	76.2

In order to preserve feasibility in a mutated chromosome a repair operation must be implemented. Each time a gene in a chromosome is mutated, the remaining genes in the chromosome are checked for infeasibility. If a command is outside the available maneuvering limits given the vehicle speed and flight path angle at the beginning of the

gene, then the command is reduced to the closest limit to the original command. This ensures that every gene in the mutated chromosome remains feasible.

3.3 Fitness evaluation

Now that the methods for creating and operating on candidate trajectories while maintaining dynamic feasibility have been established, it is important to discuss how the chromosomes created will be evaluated for performance. The fitness function defines the behavior of the genetic algorithm. The mechanics discussed in Sections 3.2.2, 3.2.3, and 3.2.4, while essential to the GA operation, make no use of any cost information. Therefore, the ability of the GA to solve a given problem depends in no small part on its ability to select good chromosomes for reproduction, which is entirely predicated by the fitness function.

Since the main goal of TERA is risk minimization, the different contributors to vehicle survivability must be modeled in the fitness function. A significant risk to low flying helicopters are MANPADs, which were introduced in Section 2.1.1. MANPAD risk is characterized by the ability of the human operator to visually detect the vehicle and then aim the rocket and fire. Assuming LOS is unobstructed, the largest factor which influences the risk is the ability of the human to pinpoint the vehicle. Anyone who has looked for a helicopter in the sky after hearing it can verify that it is not always obvious where the sound is coming from when the vehicle is a long way off, however if the source of the sound is very close it is much easier to identify it. For example, while the range of a MANPAD rocket may exceed 5000m, at less than half that range the Northrop Grumman Fire Scout autonomous rotorcraft is difficult to see and hear even when an observer knows where to look for the vehicle [36]. This argument leads to the definition of MANPAD risk at a given point in time t_j as a function of range, specifically:

$$P_{M_j} = C_{M_j} \cdot W_{R_j} \cdot L_j \qquad \text{(Eqn 3-32)}$$

where C_M is a risk weighting with value $0 \leq C_M \leq 1$, W_R gives the dependence of detectability on range, and L indicates whether clear LOS exists to the threat. In this thesis W_R is assumed to reduce linearly from a value of $W_R = 1$ at a range of 0 meters, to $W_R = 0$ at the maximum range of the MANPAD. If LOS is unobstructed to the threat, the

value of L is set to 1, otherwise the value L = 0.1 is used due to the fact detection is impossible when LOS is broken. By setting a low, but non-zero, value for L when LOS is obstructed, the GA will be rewarded for continuing to move away from the threat while out of sight. If L was set to 0 the algorithm could potentially minimize the cost by loitering out of LOS, however the threat footprint would never be escaped. In addition, threats can move, which can cause large shifts the LOS envelope of the threat, however the range envelope of the threat is only shifted equal to the distance the threat has moved. Therefore, moving the vehicle out of the threat's range before reverting to the original plan reduces the likelihood of being visually identified again due to movement of the threat.

The value of C_M in Equation 3-32 can be interpreted in different ways. One way would be to use it as a tunable weighting knob when P_M is combined with other risk measures in the fitness evaluation. Another way would be to view P_M as the probability of attrition due to the MANPAD, in which case C_M could be the probability of kill (P_k) given detection, and W_R the probability of detection given exposure to the threat during the time the vehicle has spent within the threat's LOS. While this method of attrition probability calculation simplifies the dynamics of the threat (for example, when the threat identifies the vehicle the probability of detection becomes one), a more detailed threat calculation can always be introduced by simply altering the form of Equation 3-32, due to the fact that the GA operation is irrespective of the form of the cost function.

As in many optimization problems, a conflicting interest exists which complicates the solution of the problem. It has already been established that by breaking LOS to a threat the risk to the vehicle is greatly reduced. Terrain masking can often be maximized by flying as close to the ground as possible (LOS as a function of altitude will be discussed more in Chapter 4), as even small undulations in the terrain can break LOS if both the threat and vehicle are close to the ground. The problem with this is that flying at high speeds in close proximity to the ground increases the risk to the vehicle of ground or obstacle collision. Therefore, a collision risk parameter can be introduced as the following:

$$P_C = C_C \cdot W_{CA} \cdot W_V \qquad \text{(Eqn 3-33)}$$

where C_C is the collision risk weight, and W_{CA} and W_V are risk scale factors. The variable W_{CA} provides the dependence of risk on altitude above the terrain, however to say that the risk of flying at 3 meters altitude is the same at 3 knots as at 100 knots is inaccurate. Indeed, we would like to give the GA the ability to fly at extremely low altitude if it's in the best interest of minimizing risk. The variable W_V provides the dependence on speed necessary to reduce risk at low speeds. The value of W_V is assumed to increase linearly to a value of 1 at 30 knots, and remain constant thereafter. Figure 3-22 illustrates the behavior of the combination of the two scaling factors on collision risk. Here max altitude refers to the altitude at which collision risk is deemed negligible at all speeds. For example, the max altitude would be much higher for a vehicle flying over a redwood forest as opposed to a desert. It would also be higher when flying over mountainous terrain as opposed to flatlands.

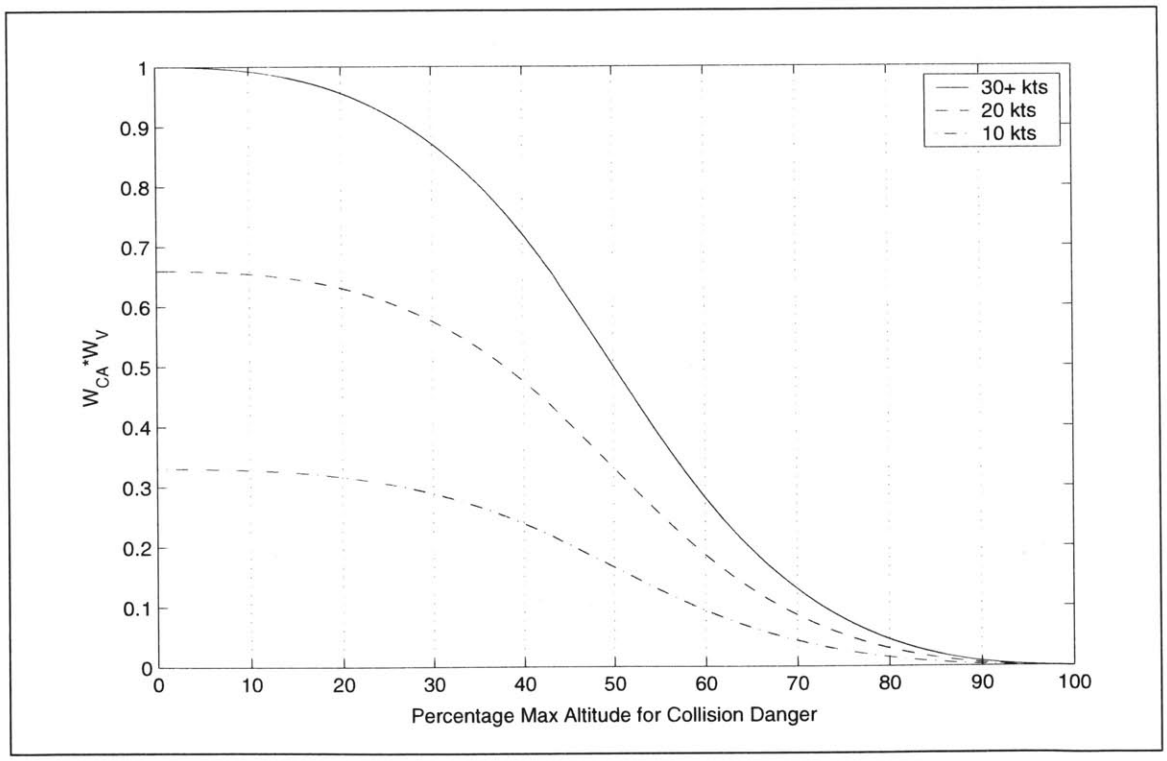

Figure 3-22 Collision risk as a function of altitude and velocity

Notice that Figure 3-22 describes the penalty induced flying above 0 meters AGL altitude. However, the potential exist for trajectories to pass through the terrain. Ground collision infeasibility is treated as a soft constraint in the fitness evaluation in order to

reduce complexity in population initialization, crossover, and mutation. Thus, the calculation of the collision cost P_C is updated in the following manner:

$$P_C = C_C \cdot W_{CA} \cdot W_V \cdot (1-b) + b \cdot W_T \cdot (-z_T) \qquad \text{(Eqn 3-34)}$$

The variable b is a logical operator that takes on value b = 1 if the point in the trajectory being evaluated is below the terrain, or b = 0 otherwise. The value z_T is the altitude of the point with respect to the terrain (z_T is negative when the point is below the terrain), and W_T is a weighting factor for varying the degree to which collision infeasibility is tolerated in the fitness. Typically the value of W_T need only be quite small (~1/10) in order to prevent trajectories that pass through the terrain from being propagated through the generations.

The goal of the TERA is solely to generate a trajectory which minimizes the risk due a pop-up threat that is within range of the vehicle. When the TERA planner is called the decision is made to forgo the current mission plan in order to increase survivability, and hence the TERA has no goal waypoint to navigate to. Instead, the vehicle is free to travel in the best direction possible in reaction to the impending threat. This means the fitness function for TERA only need take into account the risk factors discussed in the previous section.

The fitness of a given chromosome is evaluated at the end of each gene using the decoded trajectory (output space). The fitness at a specific point along a trajectory, f_i, is given by:

$$f_i = P_{M\,i} + P_{C\,i} \qquad \text{(Eqn 3-35)}$$

The total fitness of a given chromosome is found by summing the fitness values of each gene contained within the solution in order to generate a measure of the total threat exposure:

$$F = \sum_{i=1}^{\ell/\Delta t} f_i \qquad \text{(Eqn 3-36)}$$

Here the value ℓ is the time length of the chromosome, and hence $\ell/\Delta t$ determines the number of genes therein.

The value f_i is a measure of risk over the period of the preceding gene. As discussed in the previous section, the values P_M and P_C could be considered as

approximations to probability of attrition over the trajectory segment. In this case the value f_i could be expressed alternatively as a cumulative probability, i.e.:

$$f_i = P_{Mi} + P_{Ci} - P_{Mi} \cdot P_{Ci} \qquad \text{(Eqn 3-37)}$$

With f_i now representing a probability, the total probability of attrition over the trajectory can be calculated by:

$$F = 1 - \prod_{i=1}^{t/\Delta t} (1 - f_i) \qquad \text{(Eqn 3-38)}$$

While knowing the actual P_k of a solution trajectory may be of interest, Equations 3-35 and 3-36 are used for the results in this thesis for the actual fitness scoring within the GA. Although this does not provide an actual probability of attrition for a trajectory, it does provide a consistent risk metric that can allow greater variability in total fitness values between population members. Recall that the GA uses only the fitness magnitude, and no other information about the nature of the cost space. Therefore, whether the fitness is a probability or not is of no consequence. The real gain from using Equations 3-35 and 3-36 is in the wider range of total fitness possible. For example, a vehicle that encounters a pop-up threat at zero range may find that all possible trajectories have a very low probability of survival. This will result in a population all with fitness close to one, causing the GA to have poor convergence since no solutions stand out. While probabilistically speaking all solutions are in fact poor, we still require the GA to return an intuitively good solution (such as flying straight away from the threat as opposed to wandering aimlessly overhead). By summing fitness values the GA has the potential for better convergence on a purposeful solution, thus providing the required behavior from the algorithm.

3.4 Final algorithm

At this point all of the mechanisms needed for the GA have been discussed, namely the crossover and mutation operations and the fitness evaluation. The final step is to put all the elements together into the specific GA implementation for TERA response.

The operation of the GA for TERA follows very closely to the simple GA presented in Section 3.1.5. Given an initial state $\underline{x}_0 = [x\ y\ z\ V\ \Psi\ \gamma]$, the GA population is initialized to thrice the required population size (denoted by p), and each chromosome is evaluated for fitness. The initial population is then downselected to p members via a selection process called competition selection. In competition selection population members are chosen one at a time and compared to k other population members drawn at random. For each of the k members that have a higher fitness value (remember we are *minimizing* fitness as opposed to the typical GA maximization) the competing chromosome receives a score of +1. Once each chromosome has competed the top scoring p chromosomes are selected as the initial population for the GA.

The reason for generating surplus initial population members is to literally improve the quality of the initial "gene pool". Generating a large initial population and downselecting increases the number of good initial chromosomes in the GA. For this thesis a population size of p = 30 is used, and k = 20 is used for competition. The population size was chosen to balance run-time and convergence, and is typical of the population sizes seen in GA path planning [30, 35]. Higher population sizes will typically increase convergence due to the increase in genetic material, however increasing generation size directly increases computational effort per generation.

Once the initial population is found, the GA begins the generational iterations which proceed as follows: The fitness is evaluated for each chromosome in the population, and the scaled fitness values are found. Two chromosomes are then selected from the population using roulette wheel selection on the reciprocal of the scaled fitness (because of the minimization). A random number between 0 and 1 is drawn and compared to the crossover probability (p_{cross}) to determine whether crossover is performed. If crossover is selected an attempt to perform a feasible, random single-point crossover is made. Although two offspring can be produced from the crossover operation, only one offspring is required by the GA per function call, therefore each attempt actually has two chances to produce a feasible offspring. The crossover is attempted up to three times, at which point either; (1) a feasible offspring has been produced and is copied into the new population, or (2) no feasible offspring has been produced in which case the first parent chromosome is simply copied into the new population. The intent of this is to

prevent the GA from becoming stuck if no feasible crossover is possible between two chromosomes. This process is repeated until the new population has p chromosomes. In addition, the process is elitist, meaning the chromosome from the previous population with the lowest fitness is the first chromosome copied into the new generation (without crossover).

During the mutation phase each chromosome in the new population, save the first chromosome which was carried over from the previous generation, is considered for mutation. For each gene in each chromosome considered a random number is drawn from the range [0, 1] and compared to the probability of mutation value (p_{mut}). If the random number is less than p_{mut} then the gene is mutated, otherwise it is not.

Once the new population of size p has been created, the fitness of each new population member is evaluated. The iterative process of reproduction and mutation repeats until either the maximum number of generations or the allowable run time has been reached.

The question now arises as to how many genes to include in each chromosome and what time duration Δt should be associated with the genes. TERA is required to provide a trajectory that takes the vehicle out of range of the pop-up threat. This distance can vary tremendously depending on when the threat is detected, the range of the threat, the path followed out of the footprint, and the speed the path is executed. For example, evasion of a pop-up MANPAD threat with 5000 meter range could take as little as a few seconds to as much as two minutes or more depending on the circumstances of the encounter.

In response to this uncertainty, TERA implements the GA as a receding horizon controller, allowing the incremental generation of any length of trajectory with one GA formulation. In RHC a fixed time horizon (T_H) is set over which the problem at hand is solved. Only the first $T_R \leq T_H$ seconds of the solution is actually executed, however. While a given solution trajectory is being followed, the algorithm re-solves using the same horizon length with the initial condition being at time T_R along the previous horizon's solution. The previous solution from time T_R to T_H is then discarded, and the new solution beginning at T_R is executed. Figure 3-23 illustrates this concept for a

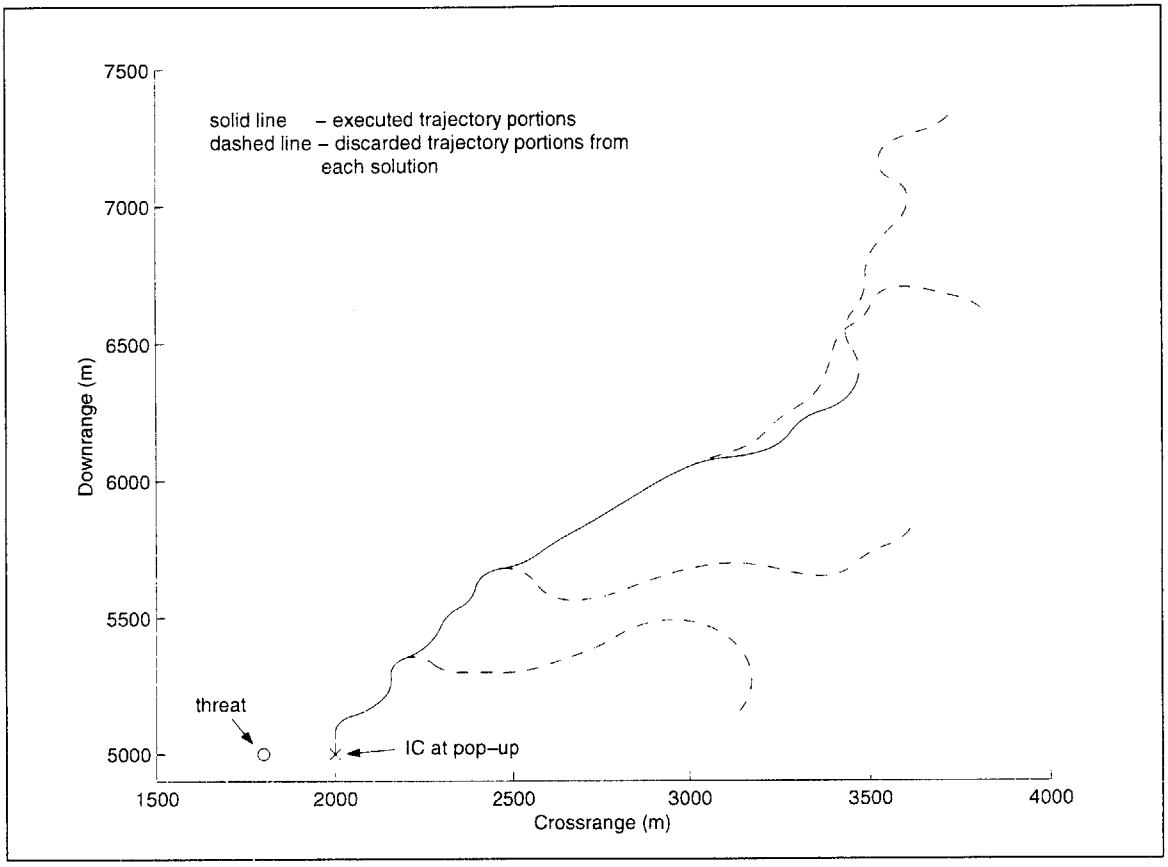

Figure 3-23 Receding horizon control example for trajectory generation

trajectory planning problem in which T_H = 30 seconds and T_R = 10 seconds. In general, increasing the solution overlap increases the algorithm's ability to react to unforeseen events; however, the value of T_R is bounded from below by the computational time required for one horizon length. Increasing T_H reduces the chance of falling into a local minimum, however it also increases the computation time, and thus T_R. Therefore, T_R and T_H allow the balancing of local minima avoidance and reaction capability with the computation time of the algorithm.

In addition to making the TERA planner adaptive to many different pop-up threat scenarios, implementing the GA in a RHC fashion is convenient for chromosome representation, wherein each chromosome is simply defined as having time length $\ell = T_H$. It was alluded to in Section 3.2.1 that the Δt for each gene would be fixed for all genes. The reason is partially due to convenience and partially due to consistency in measuring chromosome performance. The convenience aspect springs from the fact that it is most

computationally efficient to map a chromosome to the output space at the end of each gene using Equations 3-10, 3-11, and 3-12. Additionally, using fixed gene lengths ensures the fitness measure remains consistent between chromosomes in the population by summing an equal number of fitness values at equal spacing across each chromosome. Hence, using a fixed gene Δt in combination with RHC results in a GA with fixed chromosome string length equal to $\ell/\Delta t$.

With the GA operation now fully defined for TERA, the implementation of TERA proceeds in the following fashion. When a pop-up threat is encountered, TERA repeatedly applies the GA in a RHC fashion until the endpoint of a trajectory solution is beyond the range of the threat. At this point TERA turns over the command of the vehicle to the route planner that was operating at the time of the threat encounter, which must re-plan the path to avoid the new threat discovered.

Chapter 4

Threat Evasive Response Algorithm Results

In order to evaluate the effectiveness of the TERA as described in Chapter 3, several different threat encounter scenarios have been simulated. The scenarios are intended to provide realistic pop-up threat encounters representative of low altitude operation in a hostile environment, providing means to evaluate the ability of the algorithm to generate four dimensional trajectories which reduce the risk to the vehicle. This chapter presents the results of those simulations after first describing each of the threat scenarios evaluated.

4.1 Test scenarios

In the first scenario the vehicle is traveling along a predetermined trajectory through a valley formed by two surrounding hills. The vehicle is traveling at 30 m/s (~58kts) at an altitude of 30 m when a previously unknown MANPAD threat is detected after passing the side of the hill on which the threat is located. This scenario is depicted in Figure 4-1. Concentrating on the 2D figure for the time being, the original trajectory of the vehicle is shown by the solid line passing between the hills, with the direction of travel being right to left. The point at which the vehicle detects the new threat is denoted by the circle on the original trajectory, and the dotted line following shows the path the vehicle was intending to follow had the new threat not been present.

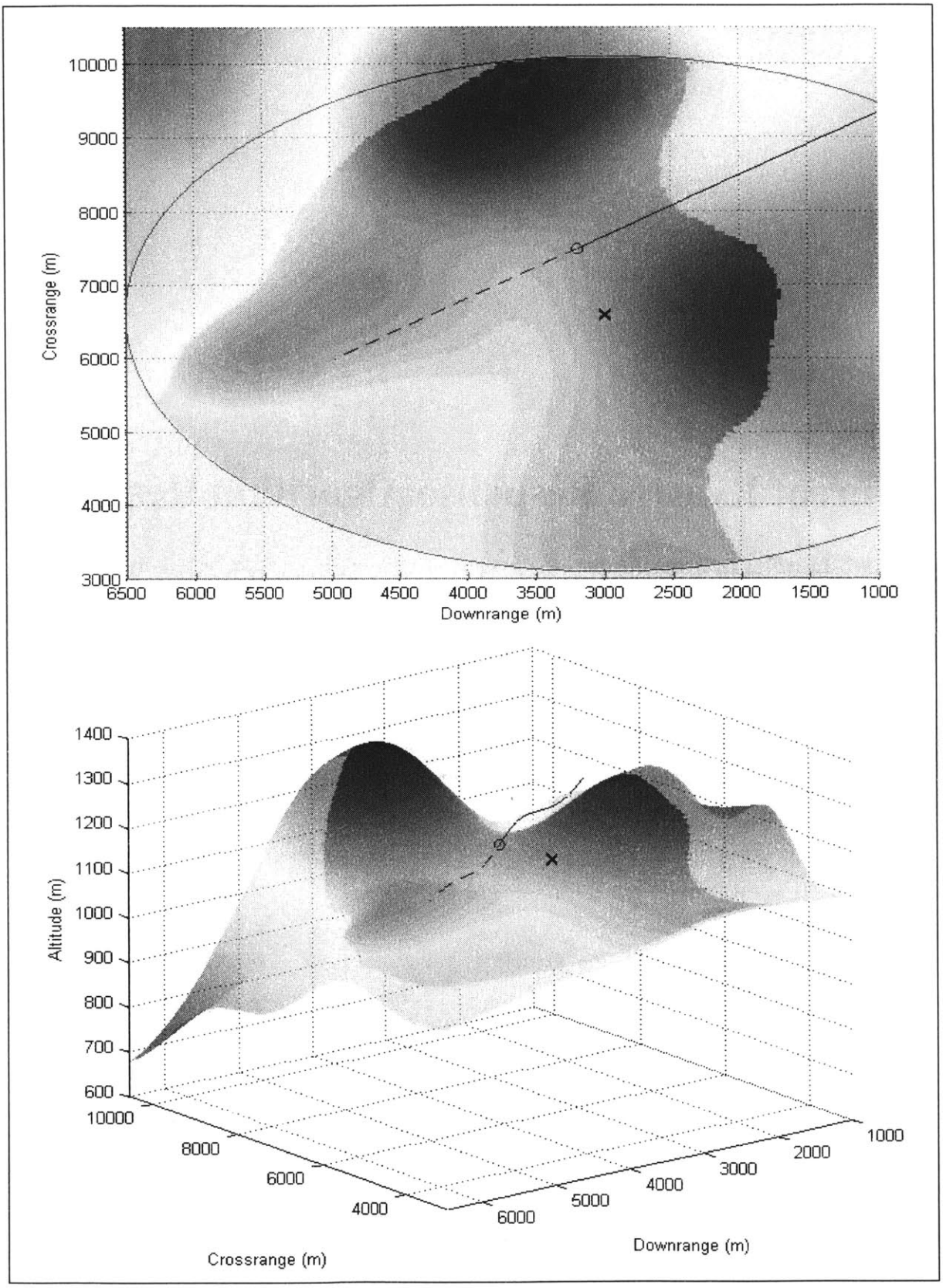

Figure 4-1 First pop-up threat scenario for TERA evaluation

The range of the threat in this case is set to 3500m (shown by circle), however due to the surrounding hills the threat is not able to see all of the terrain within the 3500 m radius. The opaque terrain region in Figure 4-1 shows the area which is visible to the threat, while the translucent terrain is out of LOS. Since the terrain being considered is three dimensional, it follows that the LOS region will be three dimensional as well. Consider the point (x_1, y_1, z_1), where z_1 is the terrain altitude at (x_1, y_1). While the point (x_1, y_1, z_1) may not be visible from a given threat location, the point $(x_1, y_1, z_1+\Delta)$, $\Delta > 0$, will always be visible for a large enough Δ. The opaque region shown thus demonstrates the region which can be seen by the threat at or above an AGL altitude of 30m (in this case Δ = 30m). Figure 4-2 shows how visibility to the threat changes with altitude, and demonstrates why flying low can have such a dramatic effect on detectability.

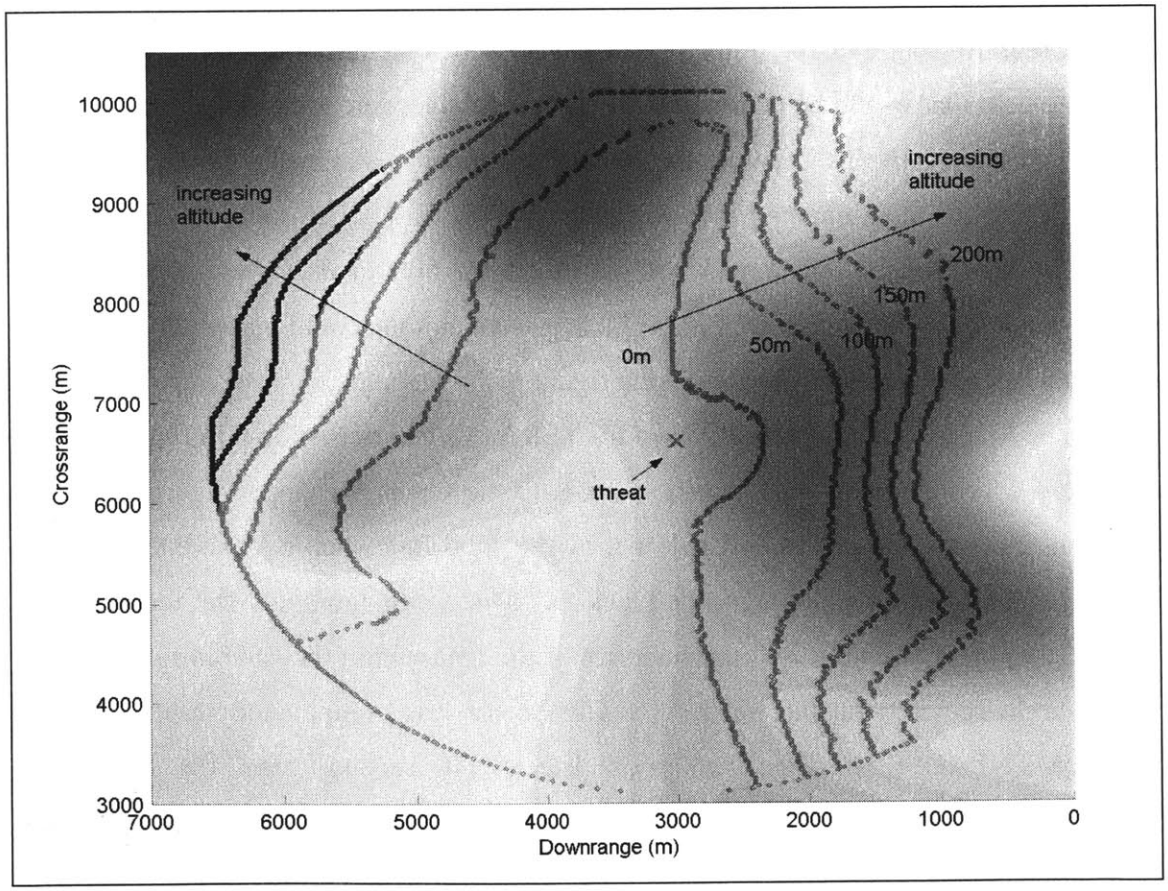

Figure 4-2 Visibility footprint of threat in Scenario #1 for a vehicle at different altitudes

The second scenario was selected due to the pronounced effect in the area observable by the threat depending on what altitude the vehicle is flying at. The vehicle is

initially flying right to left in the 2D plot of Figure 4-3 when a new threat at the location denoted by the "X" is detected. The detection occurs when the vehicle is at the point shown by the circle, while traveling at 30 m/s and 30m AGL altitude. The threat's engagement radius is shown by the black circle. The opaque region within the circle represents the area in which a vehicle flying at 20m AGL can be seen, while the darker translucent area shows the area seen at 60m AGL. In this case the vehicle could potentially exit the threat radius completely out of LOS by making intelligent altitude corrections.

The third threat scenario evaluated is depicted in Figure 4-4. The vehicle is initially traveling at 30 m/s and 30m altitude along the nominal threat free trajectory depicted as the straight black line, with the direction of travel being right to left. At the point denoted by the circle on the nominal trajectory a new threat located at the 'X' is detected requiring an evasive trajectory re-plan. As before, the continuing dashed line shows the path that would be continued upon if the vehicle were incapable of re-planning to avoid the threat. In this example the MANPAD threat is assigned a range of 5000m, giving the threat a commanding view of the expansive valley below it. In this case the vehicle would be in complete view of the threat for 5000m if it were to follow the original trajectory. The area in which a vehicle at 30m altitude could be seen by the threat is denoted by the opaque terrain area in the figure.

The fourth and final scenario used for TERA evaluation is shown in Figure 4-5. In this case the vehicle is flying from the bottom to the top of the upper figure along the initial trajectory (30m/s, 30m AGL) shown by the solid line when a MANPAD threat is detected at the location denoted by the black 'X'. Due to the relatively flat terrain in the vicinity, the threat is able to see the majority of the area within the 4500m range shown by the circle centered on the black 'X'. This scenario is complicated when a second pop-up MANPAD with range 5000m is detected 40 seconds after the first threat detection, requiring the TERA to avoid both threats. The second threat is located at the white 'X', and the range is depicted by the circle centered on the threat. As in the previous figures, the area in which the combined threats can see a vehicle at 30m AGL is represented by the opaque terrain region.

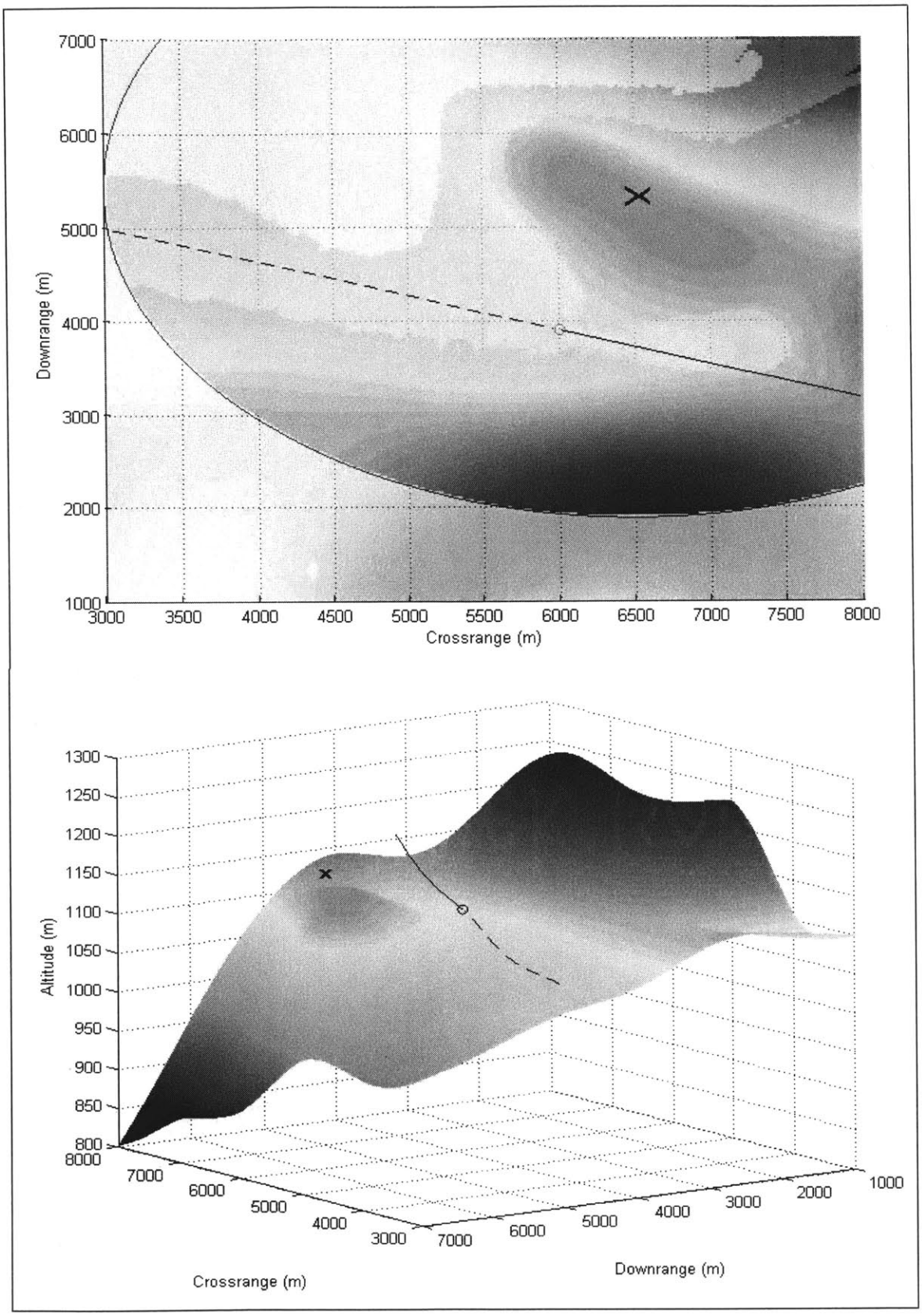

Figure 4-3 Scenario #2 for TERA evaluation

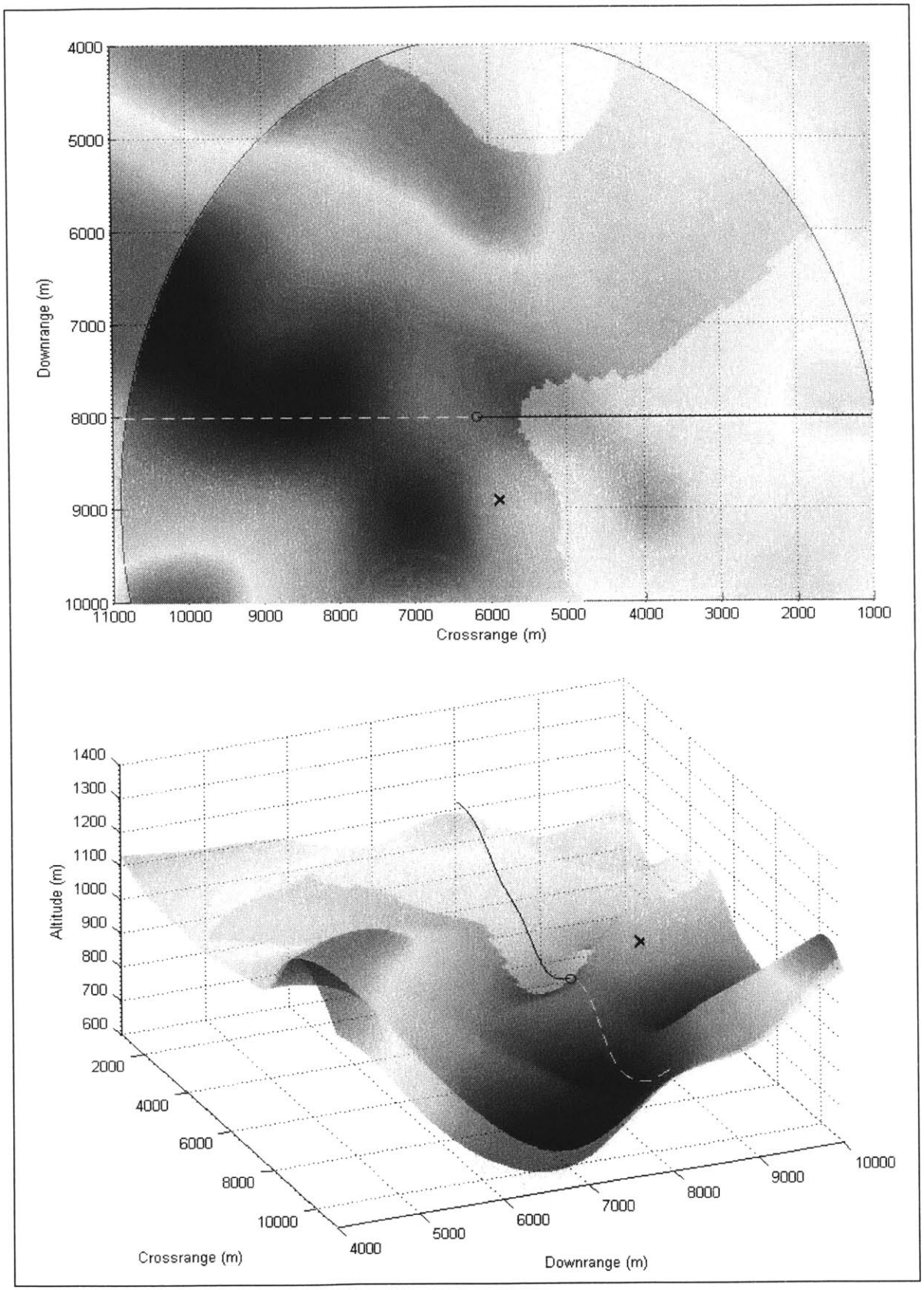

Figure 4-4 Scenario #3 for TERA evaluation

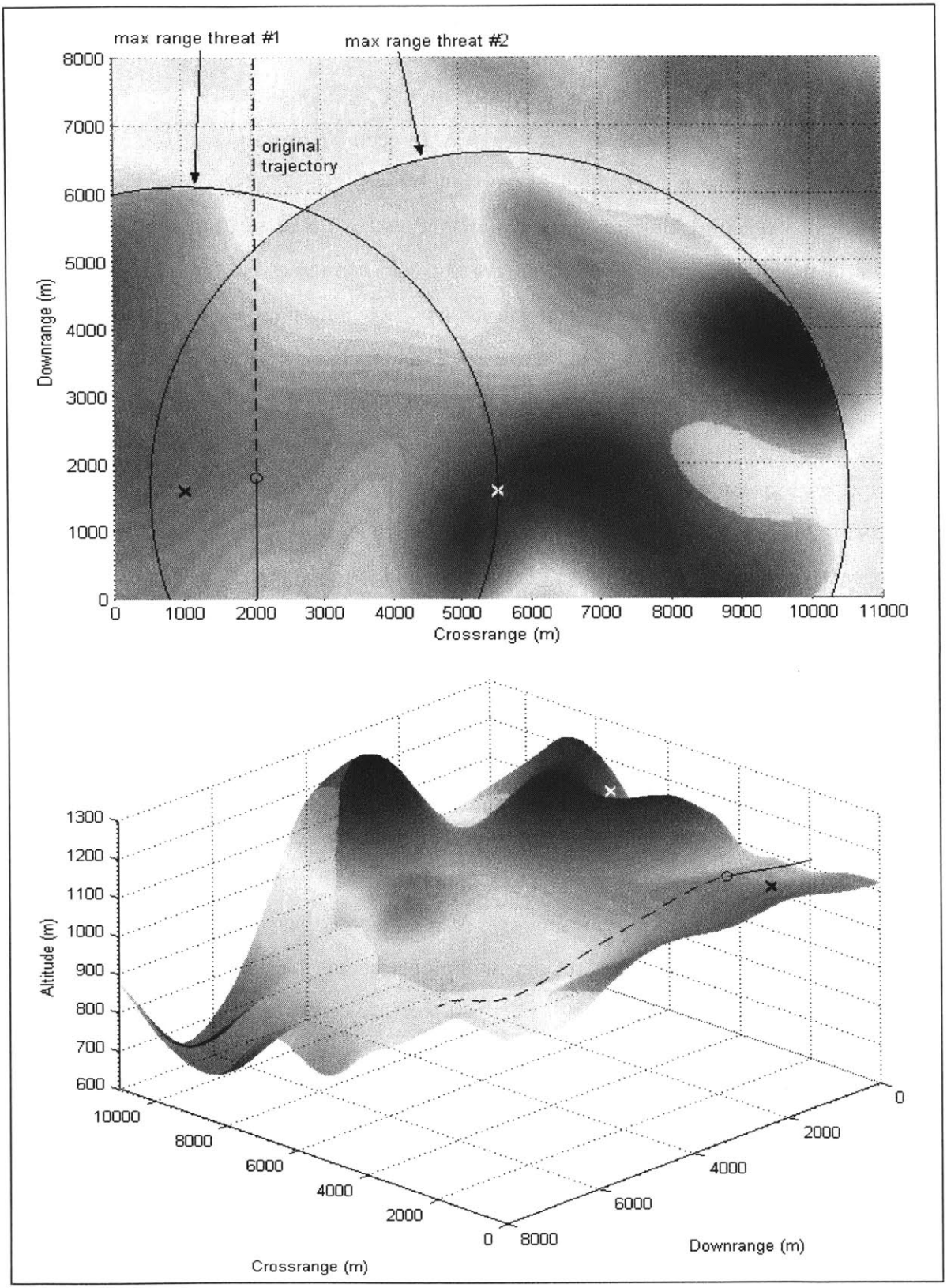

Figure 4-5 Scenario #4 for TERA evaluation

4.2 Simulation results

For each of the scenarios discussed in the previous section the TERA was simulated in order to evaluate its effectiveness. In each circumstance a solution horizon of T_H = 40 seconds was used, with a re-plan horizon of T_R = 20 seconds. The GA was executed in a receding horizon fashion until the end point of a returned solution was outside of the radius of the threat to be avoided (or both threats in the case of scenario #4). The GA iterated for 600 generations for each solution, used a probability of crossover of 0.5, a probability of mutation of 0.1, and a gene duration of 2 seconds.

The first scenario was meant to evaluate the ability of the TERA at seeking out globally "good" areas to fly in a setting where the generally correct solution is intuitive to the human observer. The overhead view in Figure 4-1 shows that LOS to the threat can be quickly broken if the vehicle were to double back into the valley from which it came, significantly reducing exposure over continuing on its original path. Once LOS is broken, according to the cost function the best course of action is to fly radially from the threat until out of range.

Figure 4-6 shows the result of 15 independent TERA evaluations in response to the first scenario. Indeed, the (x, y) path behavior in this scenario is intuitive, as the vehicle immediately performs a 180° turn and backtracks until out of LOS. Once LOS break has been achieved all 15 trajectories turn and radially exit the range of the threat.

Figure 4-7 shows a single trajectory out of the group of TERA solutions plotted in Figure 4-6, with each marker representing a Δt = 2 seconds. The endpoint of each gene is marked with either an '*' or an 'o', depending on whether the point on the trajectory is within LOS or without, respectively. This particular trajectory is able to reduce the time of threat exposure after the initial detection to 14 seconds, which is well below the reload and fire rate of a MANPAD device. Figure 4-7 also shows a white line emanating from the MANPAD location which illustrates the behavior of the trajectory after LOS has been broken.

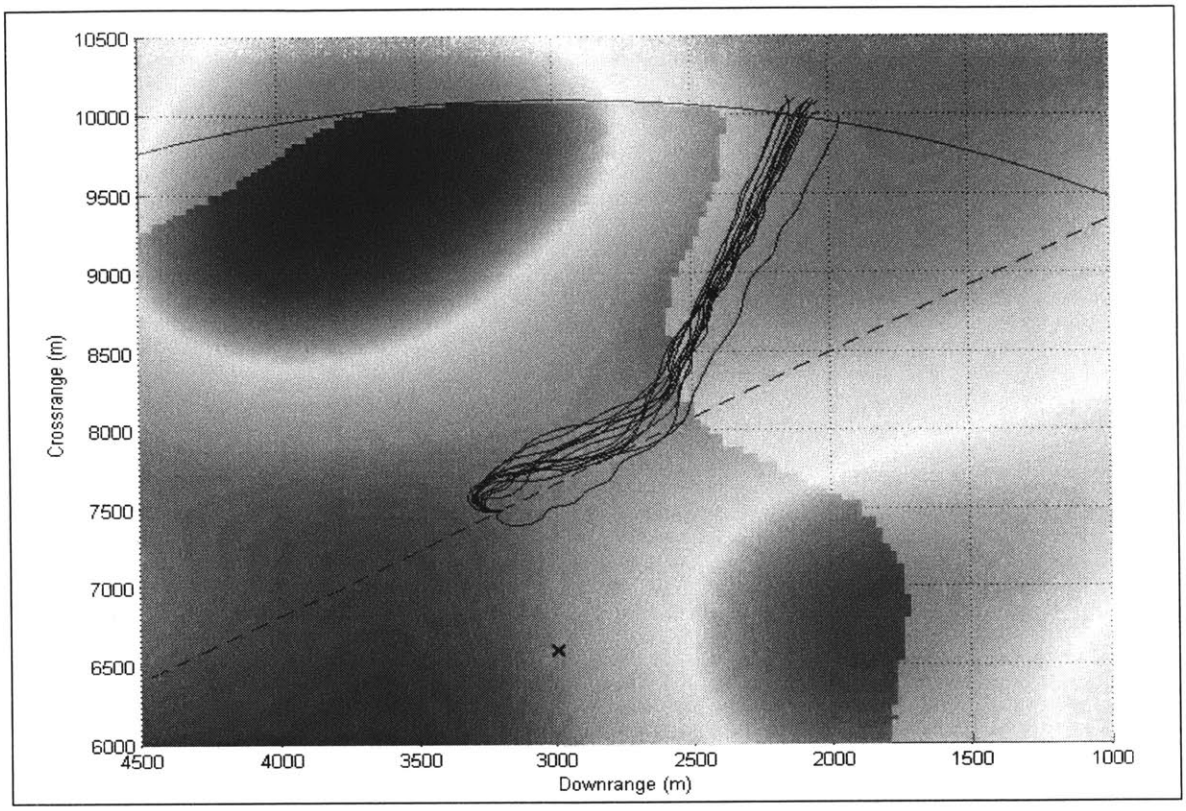

Figure 4-6 Results of multiple TERA evaluations of Scenario #1

While Figure 4-6 appeases the intuition of what the general path behavior should look like, the question remains as to whether TERA is making good use of altitude and velocity for risk reduction. Indeed, it is the addition of altitude and velocity which makes the TERA problem far more difficult than 2D path planning, and which in part motivates the use of a GA in the first place. Figure 4-8 begins to answer this question by showing the AGL altitude behavior (right column) of five of the fifteen trajectories from Figure 4-6. For these results an altitude of 40m was set as the maximum obstacle risk altitude referred to in Figure 3-22, and denoted in Figure 4-8 by the "min obstacle free" line. The "min LOS" line shows the AGL altitude above which LOS is clear at a given (x, y) location along the trajectory. The typical behavior of TERA is to fly above the 40m line so as to impose no obstacle risk on the vehicle, which is evident in the initial increase in AGL altitude in several of the trajectories, and also in the increase in AGL in the latter half of each trajectory after LOS is broken. However, the TERA appears to be diving the trajectories below 40m in order to get under the minimum LOS altitude, sacrificing

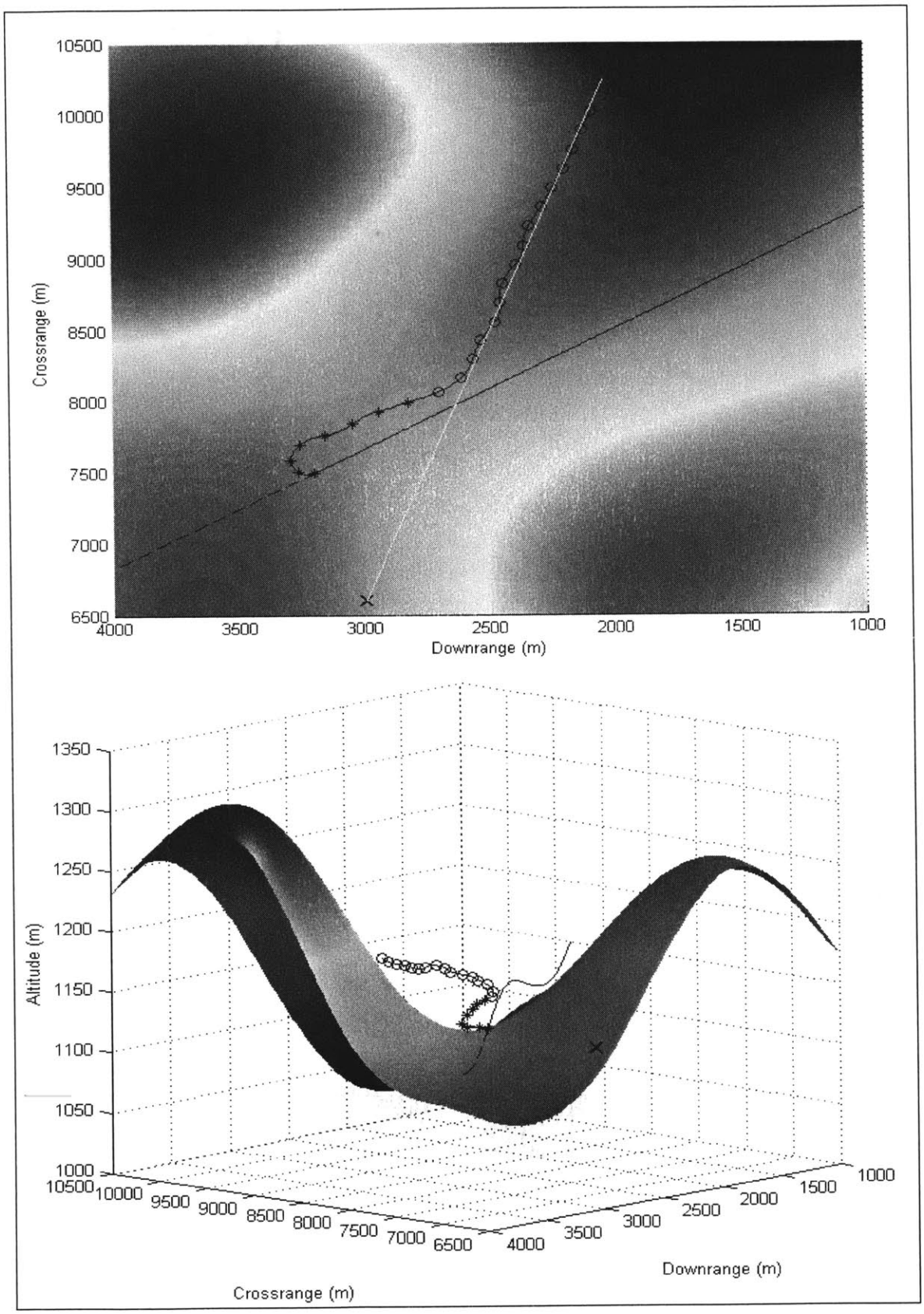

Figure 4-7 Single TERA result for Scenario #1

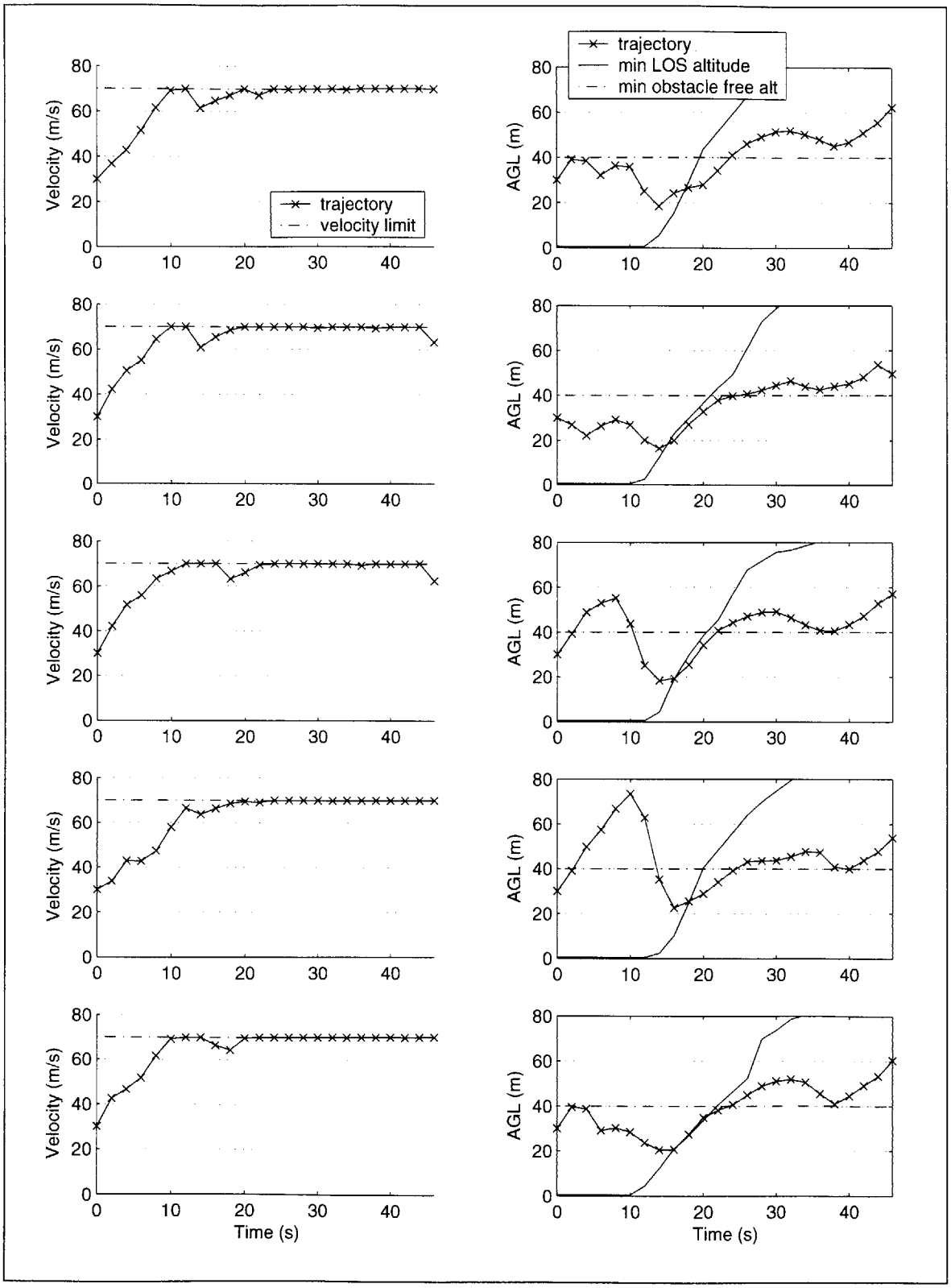

Figure 4-8 Velocity and altitude behavior for trajectories in Scenario #1

obstacle risk to reduce LOS exposure. In fact, in three of the five cases the trajectory nearly perfectly tracks the LOS line for a short duration.

Figure 4-8 also shows the velocity behavior of the same five trajectories in the left column. In each case the vehicle accelerates to and tracks (for the most part) the maximum velocity for the duration of the trajectory, allowing the vehicle to minimize the time required to follow the path out of threat range. Note that each trajectory has a slight dip in velocity around the 15 second mark. This is because the vehicle begins to climb once under the LOS break, however at maximum velocity the ROC limit is zero (see Figure 3-9). The vehicle must decelerate slightly to increase the ROC limit in order to allow the climb to take place.

The TERA response to the first scenario suggests the algorithm is indeed using all 4 dimensions of the trajectory space for risk reduction; however, the reduction of time in LOS shown in Figure 4-8 over flight at the 40m obstacle safe altitude is rather small. This raises the question of whether the algorithm would truly be able to take advantage of altitude in a setting where reducing altitude would have a more profound effect of risk. To analyze this question, the GA that is the core of TERA was used to generate multiple solutions for the first 40 second time horizon following threat detection in Scenario #2. The analysis of the altitude behavior of several of the resulting trajectories is shown in Figure 4-9, and in fact, the altitude behavior is similar to that seen in the first scenario. The TERA proves to be quite capable of altering altitude in order to reduce threat risk by breaking LOS. While this does not prove that including altitude in the trajectory search space will always be fruitful (if the terrain was completely flat altitude would have negligible effect), it does show that the GA implementation is capable of significantly reducing threat risk via altitude given the proper threat encounter scenario.

The third scenario is similar to the first in that it is intuitive that the correct TERA result will double back on the initial trajectory in order to fly into the LOS shadow behind the hill on which the threat is perched (Figure 4-4). Indeed, the results of 15 independent TERA evaluations, shown in Figure 4-10, demonstrate the algorithm reaches the same conclusion as all solutions quickly turn and head for the shadow region. Had the vehicle followed the original trajectory it would have been exposed to the threat for over 160 sec,

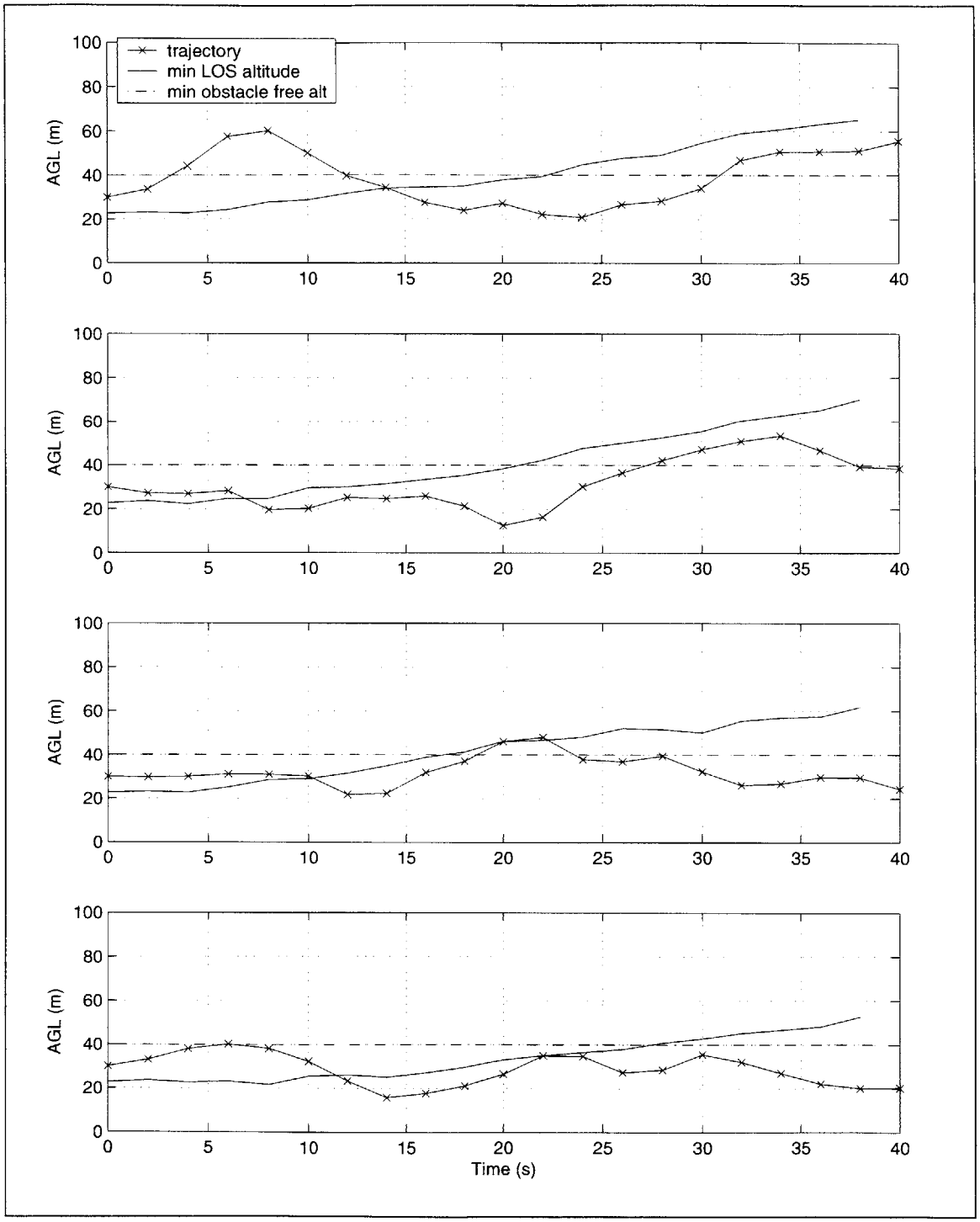

Figure 4-9 Reduction in LOS exposure in Scenario #2 by varying AGL

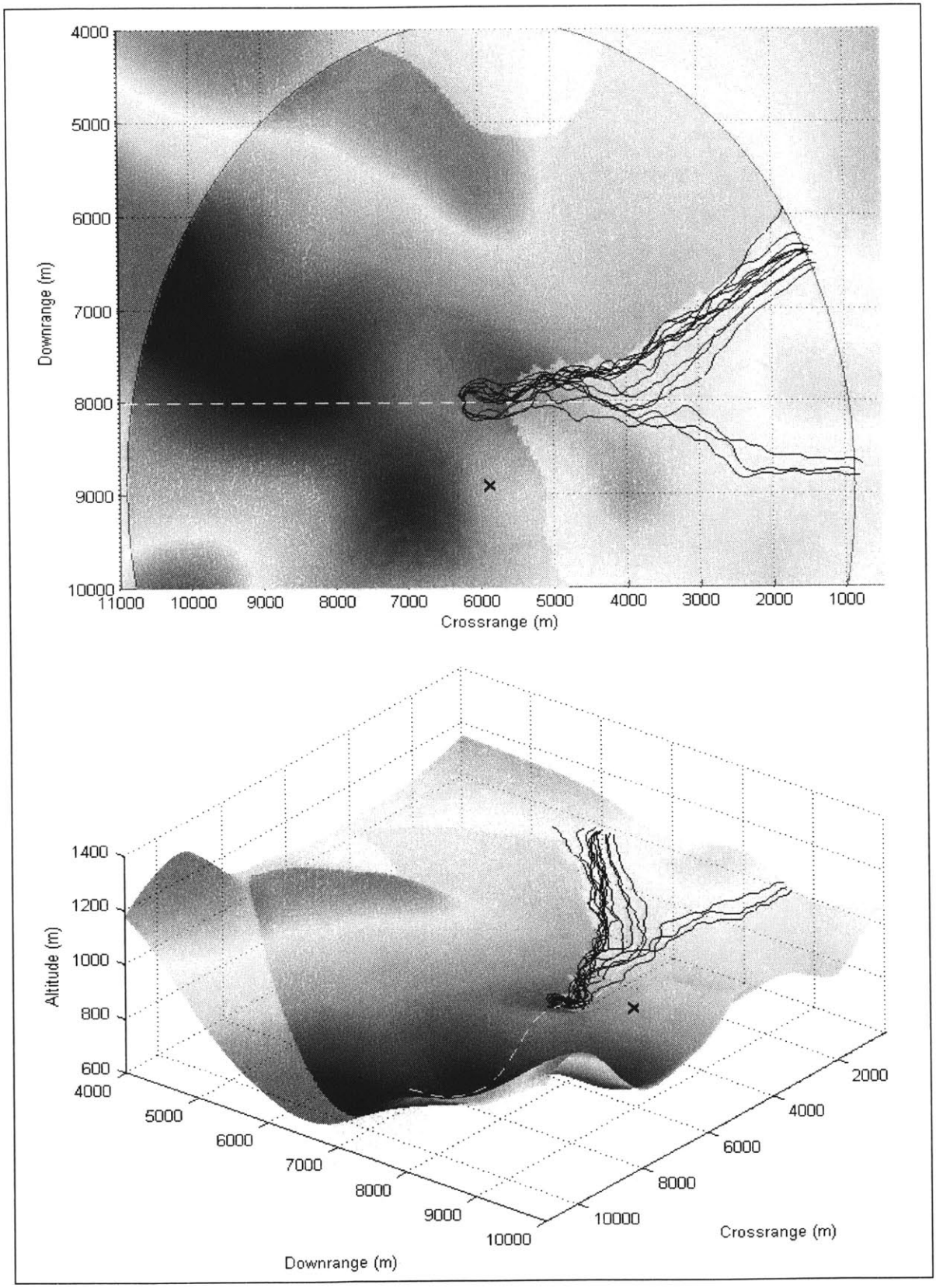

Figure 4-10 Multiple TERA responses to Scenario #3

however, the average time exposure after detection of the threat for the 15 cases shown was reduced to only 14 seconds.

Taking a closer look, the trajectories shown in Figure 4-10 appear to be kinkier than those in Figure 4-6. It is also curious that 3 of the trajectories split off from the other 12 which follow a more radial pattern once LOS has been broken. Both of these anomalies turn out to be caused by the dynamic constraints on the vehicle. Figure 4-11 shows the flight path angle and velocity histories of four of these trajectories. During the first 20 seconds γ tends to spike as the trajectory attempts to fly down into and then out of a bowl shaped feature in the terrain. After this the vehicle must fly uphill out of threat range (Figure 4-10). The climb effectively limits the velocity such that the required γ can be achieved, a result of the fact that at maximum velocity the ROC limit is zero. This also appears to explain the wandering in the trajectories, which is the greatest in the 3 outlying trajectories. By splitting to the right the trajectories are able to reduce the gradient of the terrain they are attempting to track, hence increasing the allowable velocity and escaping the threat faster. Likewise, the kinky behavior is likely caused by the need to reduce the slope of the climb due to the rate of climb limit on the vehicle. The kinks in the trajectories are similar in nature to switchbacks in a road or trail which, while increasing distance, serve to reduce the gradient of the path.

In order to assess the ability of TERA to maximally exploit the vehicle capability while still honoring the dynamic constraints, the same scenario was evaluated after increasing the ROC limit as shown in Figure 4-12. The 15 resulting trajectories are shown in Figure 4-13. While increase in ROC occurred at all velocities, the most significant change is the allowance of non-zero ROC at the maximum velocity. The flight path and velocity profiles of 4 of the resulting trajectories are shown in Figure 4-14. The new γ limit at maximum velocity turns out to be enough for the vehicle to climb out of the valley, resulting in trajectories that are much closer to the velocity limit and slightly smoother in appearance. Note that none of the trajectories in this case split to the right of the hill on the edge of the threat range since contour following is not required with the increase in the ROC limit. This example demonstrates that TERA is in fact taking advantage of the maneuvering capability of the vehicle for risk reduction while operating with in the dynamic constraints imposed.

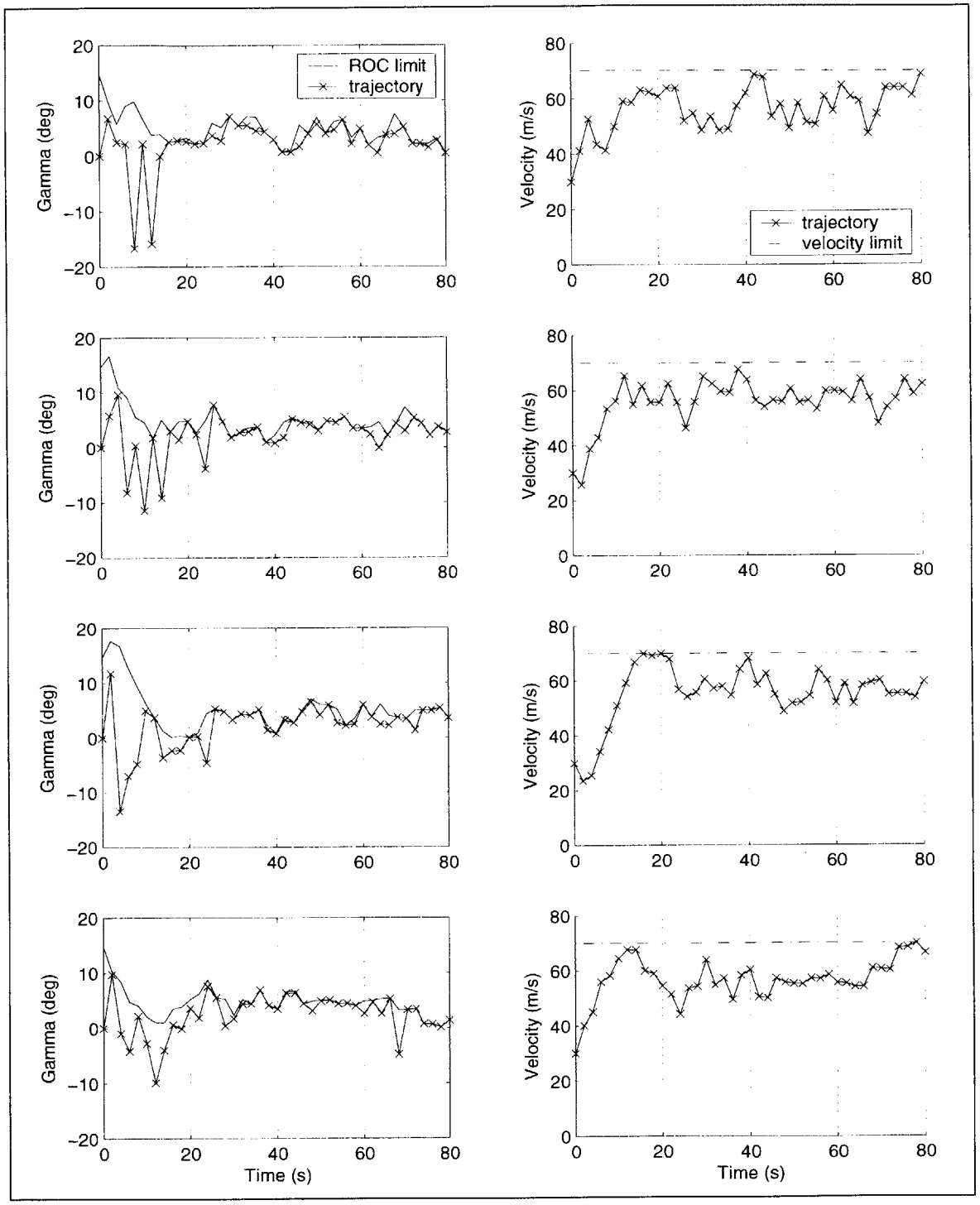

Figure 4-11 Flight path angle and velocity histories for Scenario #3 results

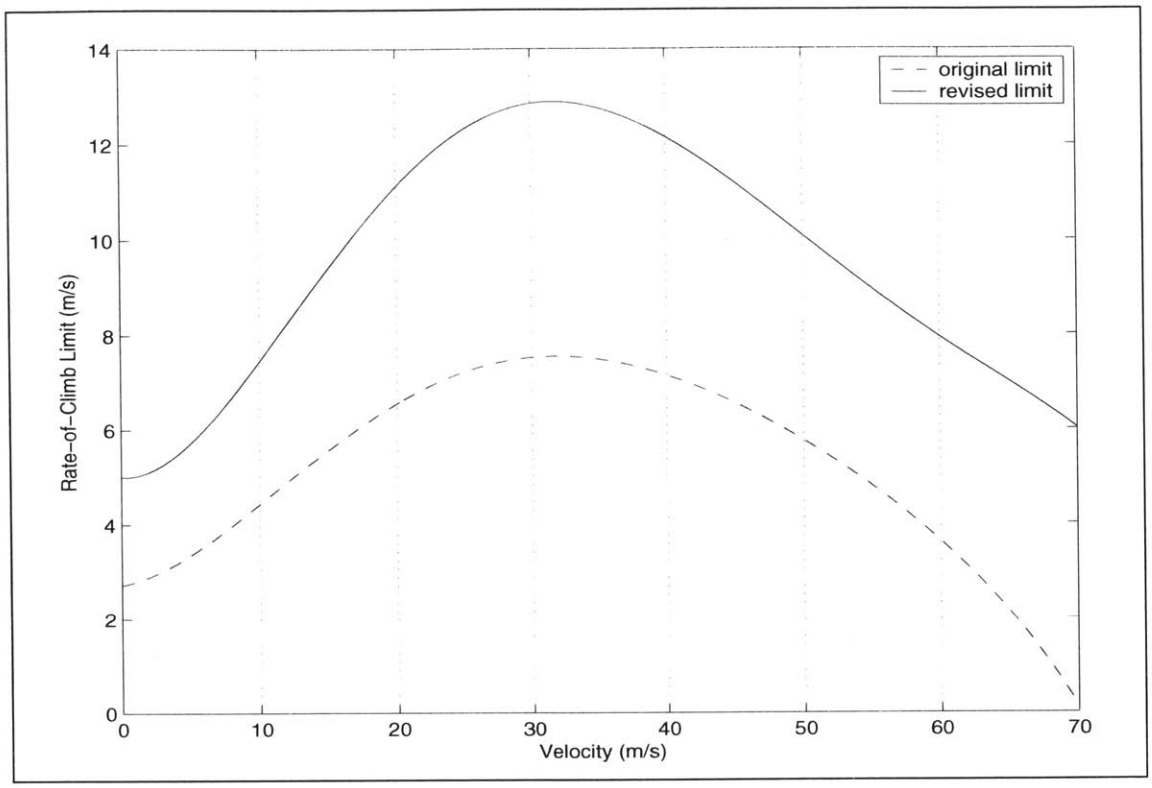

Figure 4-12 Revised rate of climb limit for evaluation of dynamic limit sensitivity

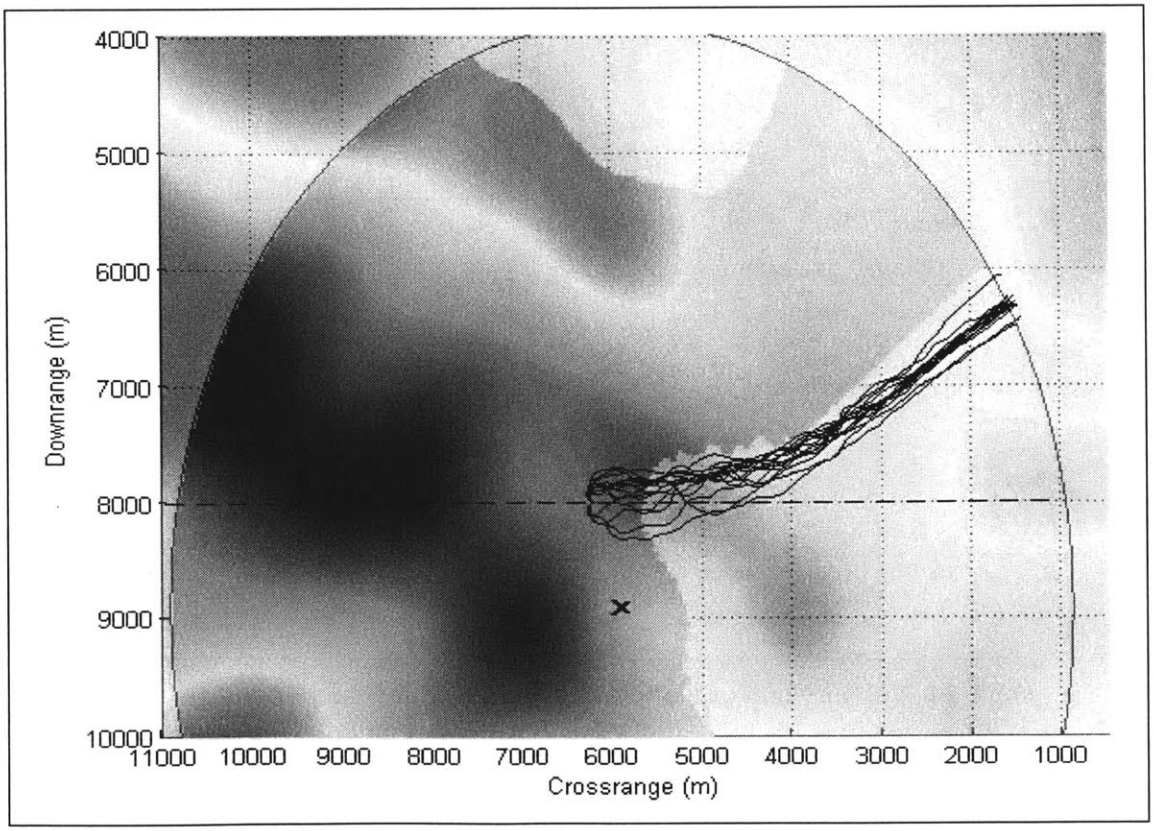

Figure 4-13 Scenario #3 results with revised rate of climb limits

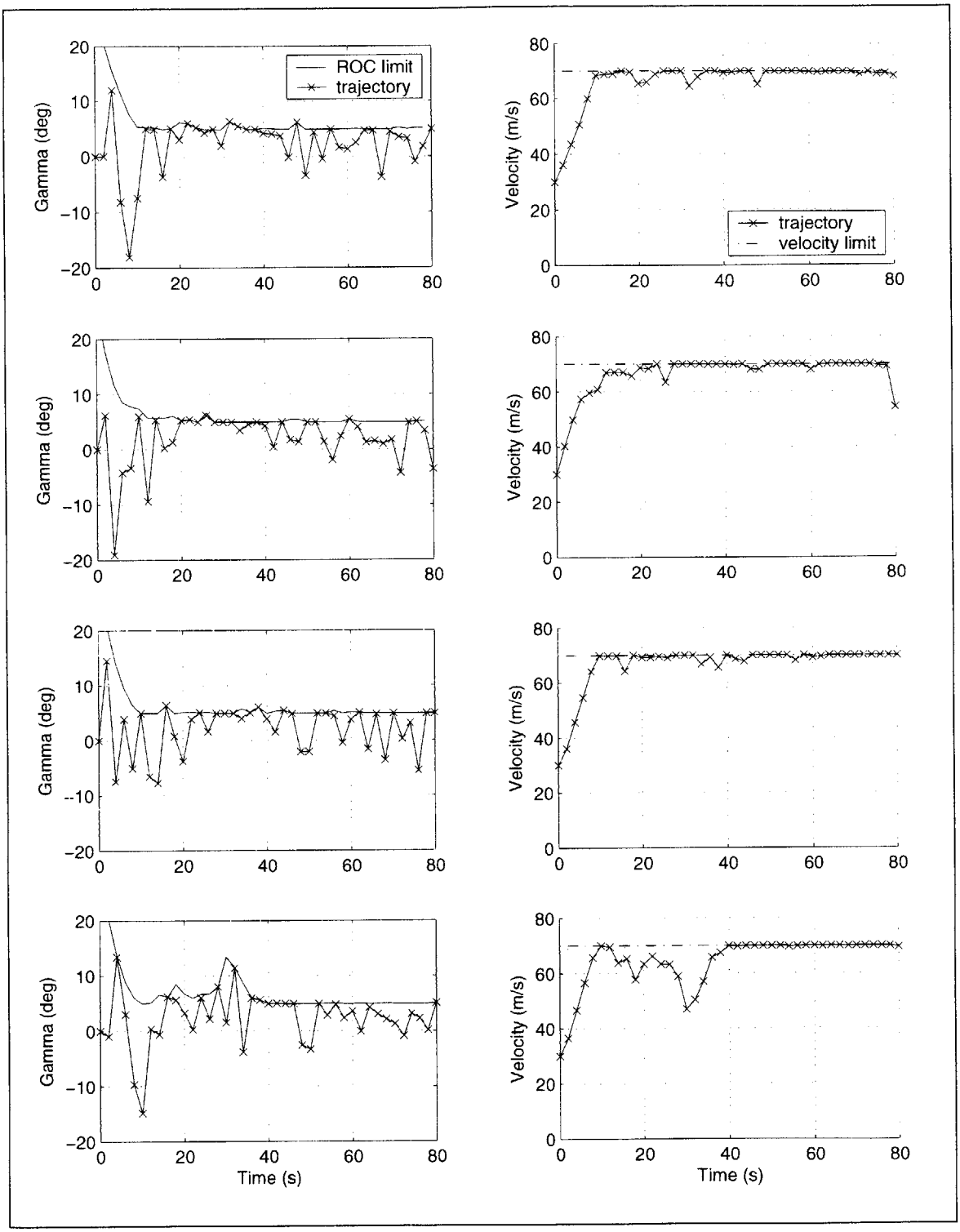

Figure 4-14 Velocity and rate of climb histories for revised limit trajectories

The fourth scenario differs from the previous three in that no major LOS shadow area exists for the TERA to use for reduction of exposure, however, it does serve to show

the ease of which multiple threats can be dealt with by TERA. Figure 4-15 shows 15 resulting trajectories from TERA for this scenario. The white line emanating from the first threat (black 'X') shows that the initial response to the threat is to flee the scene in the most direct sense possible, due to the total LOS coverage of the threat. The second threat (white 'X') is detected 40 seconds after the first threat detection, upon which the trajectories perform a left turn to avoid the new threat as well. While the trajectories continue to move away from each threat after the second detection, the path behavior is dominated by the influence of the newer threat due to the closer proximity. Once the range of the first threat has been exceeded the trajectories turn to exit the second threat's range radially (as indicated by the two dashed lines coming from the second threat).

Since the vehicle remains in LOS for the duration of the trajectory it behooves the vehicle to fly at the maximum velocity in order to evacuate the area as quickly as possible. The TERA does an excellent job of this, which is evident from the velocity profiles shown in Figure 4-16. Following the initial acceleration (the first 8 seconds), the average velocity over all 15 trajectories is 69.83 m/s (the max velocity limit is 70 m/s) with a standard deviation of 0.40 m/s.

Figure 4-16 likewise shows that the altitude behavior is intuitive based upon the particular fitness function used. Recall that the obstacle/terrain risk metric in the fitness evaluation penalizes flight under a minimum safe AGL altitude (40m in this case). In addition, a small linearly increasing cost was included to penalize flight above 80m, which was included to motivate terrain tracking (otherwise there would be no cost associated with flying at any altitude over 40m). The second window in Figure 4-16 shows that all trajectories are in fact limited between 40-80m following the climb from the initial AGL altitude of 30m. The correlation in AGL altitude is explained by the third window of the figure, which shows the absolute altitude behavior of all 15 trajectories along with the terrain elevations encountered along each trajectory. The trajectories effectively smooth the terrain due to the fact that the algorithm has no reason to track each contour in the terrain as long as the AGL stays within the 40-80m cost free zone, leading the wavy behavior in the AGL. For example, the dip in AGL seen at ~43 seconds corresponds to a rise in the terrain which TERA chooses not to track because it has no ill effect on the fitness.

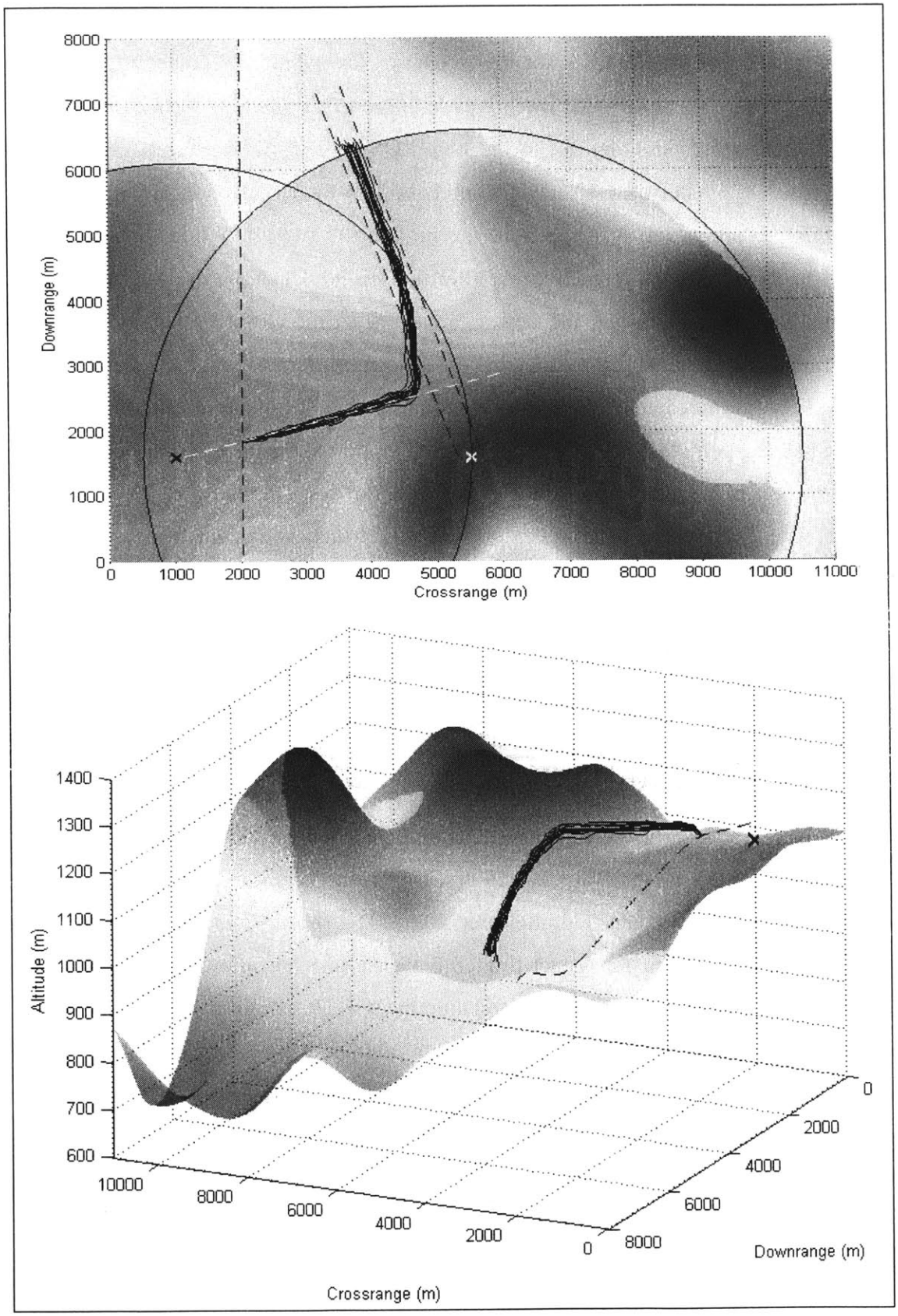

Figure 4-15 Multiple TERA results for Scenario #4

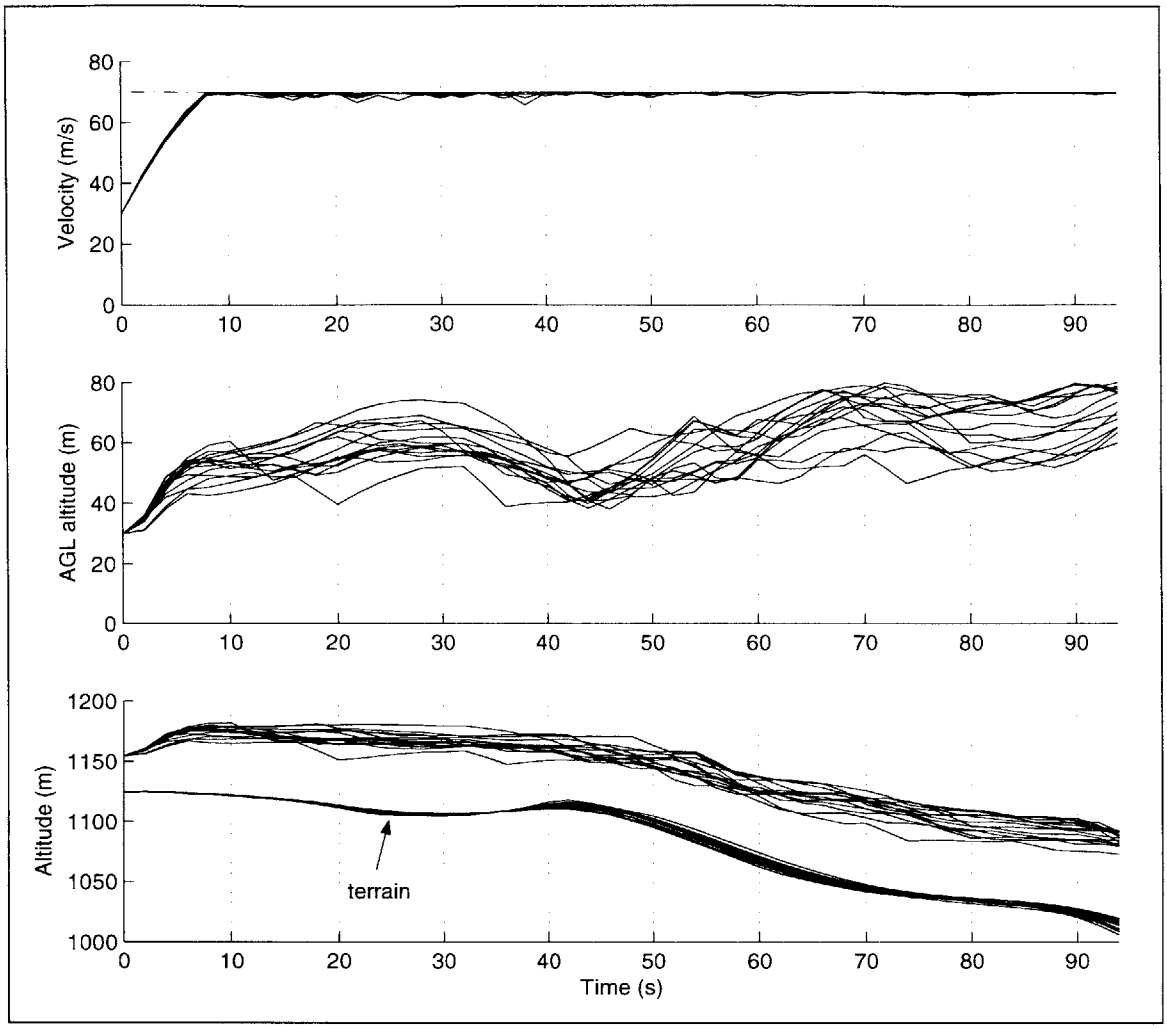

Figure 4-16 Velocity and altitude behavior of Scenario #4 TERA results

Thus far it is clear that TERA is capable of taking full advantage of all four dimensions in the trajectory space in a way befitting of cost reduction. While the altitude behavior shown in Figure 4-16 is indeed intuitive, it is not clear whether the algorithm would perform as well if the range of risk free altitudes were to be restricted in order to impose a lower operational ceiling. In order to further assess the ability of TERA to control altitude the fourth scenario was reevaluated after updating the obstacle risk cost as shown in Figure 4-17. The linear cost increase at higher altitudes represents the artificial penalty imposed in order to reduce the likelihood of the vehicle flying at arbitrarily large altitudes. By altering the slope of the linear segment the "hardness" of the operational ceiling is defined. The resulting 15 trajectories from the TERA evaluation using the updated obstacle risk cost are shown in Figure 4-18.

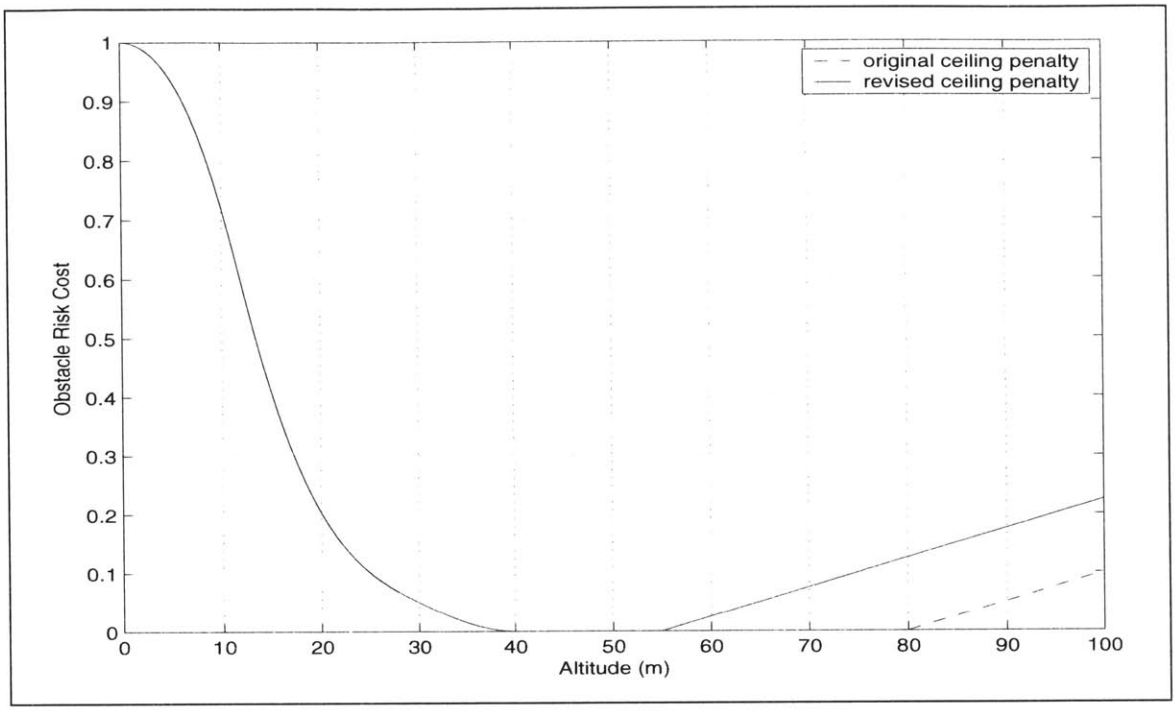

Figure 4-17 Updated obstacle risk function for improved terrain following

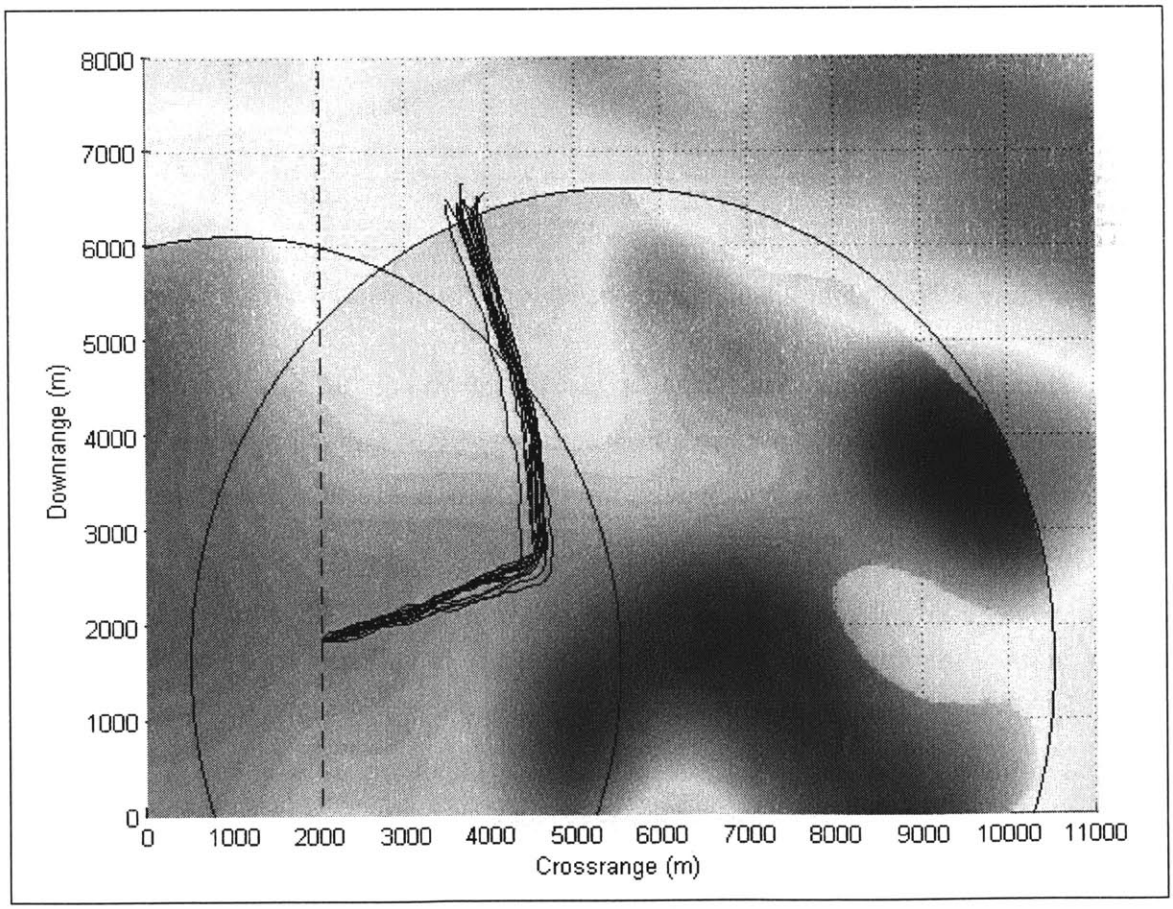

Figure 4-18 Scenario #4 TERA results with updated obstacle risk

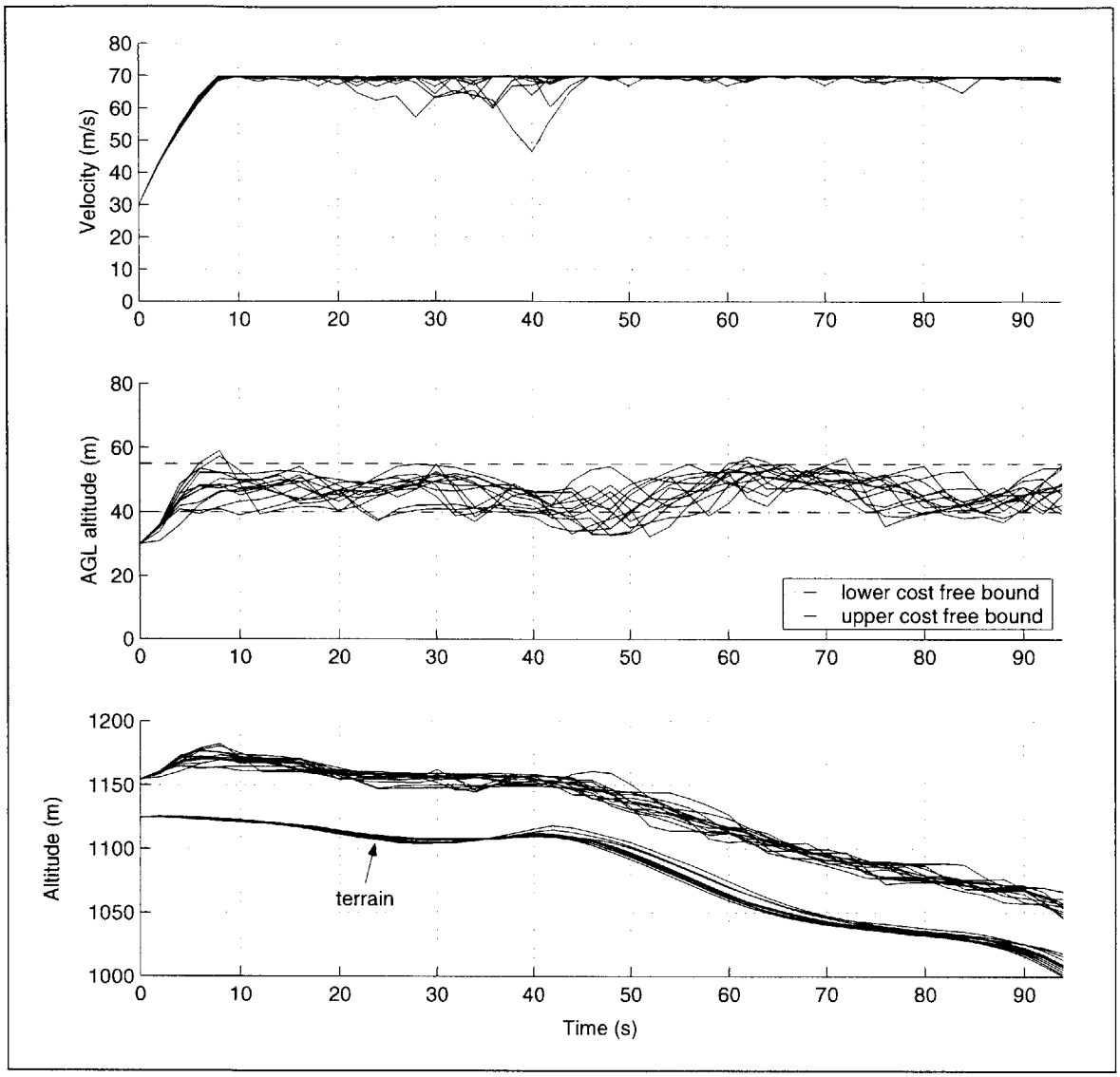

Figure 4-19 Velocity and altitude profiles for Scenario #4 results with updated obstacle risk

Likewise, the velocity and altitude profiles of the TERA results are shown in Figure 4-19. The trajectories do a quite reasonable job of remaining within the 40-55m obstacle risk free zone, with the upper bound broken only a few times over all the trajectories. The larger deviation below the lower bound at ~48 seconds is due the rise in terrain, which the algorithm ignores while the AGL remains over 40m. The algorithm then pulls up once the trajectory starts to impose obstacle risk, however the small ROC limit available due to the high velocity reduces the ability to recover altitude quickly. The increase in cost due to this deviation was low enough as to not warrant velocity reduction

in order to achieve the higher climb rates required to reduce the undershoot in AGL. More velocity fluctuation was present over that seen in Figure 4-16, however, showing that the more stringent altitude window required more positive flight path angle commands, leading to the need for reduction in velocity.

Since GAs (and hence TERA) are not deterministic, the issue of repeatability of results must be addressed. Repeatability can be of interest for several reasons, among them being validation of flight software. The simulation results presented so far suggest a high degree of repeatability in the results, as opposed to widely varying solutions. Figure 4-6, Figure 4-10, and Figure 4-15 clearly show clustering in the results, with the Scenario #3 results experiencing the greatest variation. In order to further assess the behavior of TERA, Scenario #3 was re-evaluated 150 times independently, with the results shown in Figure 4-20. In this case 12 of the 150 resulting solutions split from the rest in order to follow the terrain contours to the right of the hill on the edge of the threat's range. However these 12 trajectories are not a set of spurious and inferior results; rather, with the mean fitness of the 12 outlying trajectories being 10.00, compared to the mean fitness of 10.35 of the entire group, these outliers prove to be competitive alternatives to the other trajectories. It is the clear ability of the solutions in Figure 4-20 to seek out the area of reduced LOS which is of importance in evaluating the TERA, and in this respect the algorithm shows a high degree of repeatability.

4.2.1 Run-time considerations

Inherent in the role of the threat evasive response algorithm is the need for real-time operability. The genetic algorithm discussed in Chapter 3 was implemented in Matlab Release 13, and used to generate all results contained within this thesis. Because Matlab functions are interpreted as opposed to compiled, the resulting run-times of the algorithm are not representative of what would be expected from a complied language such as C or Fortran. Although any statements about the real-time capability of the current TERA are speculative, there exist several pieces of evidence which build a strong argument that the GA based TERA presented in this thesis is fully capable of real-time operation.

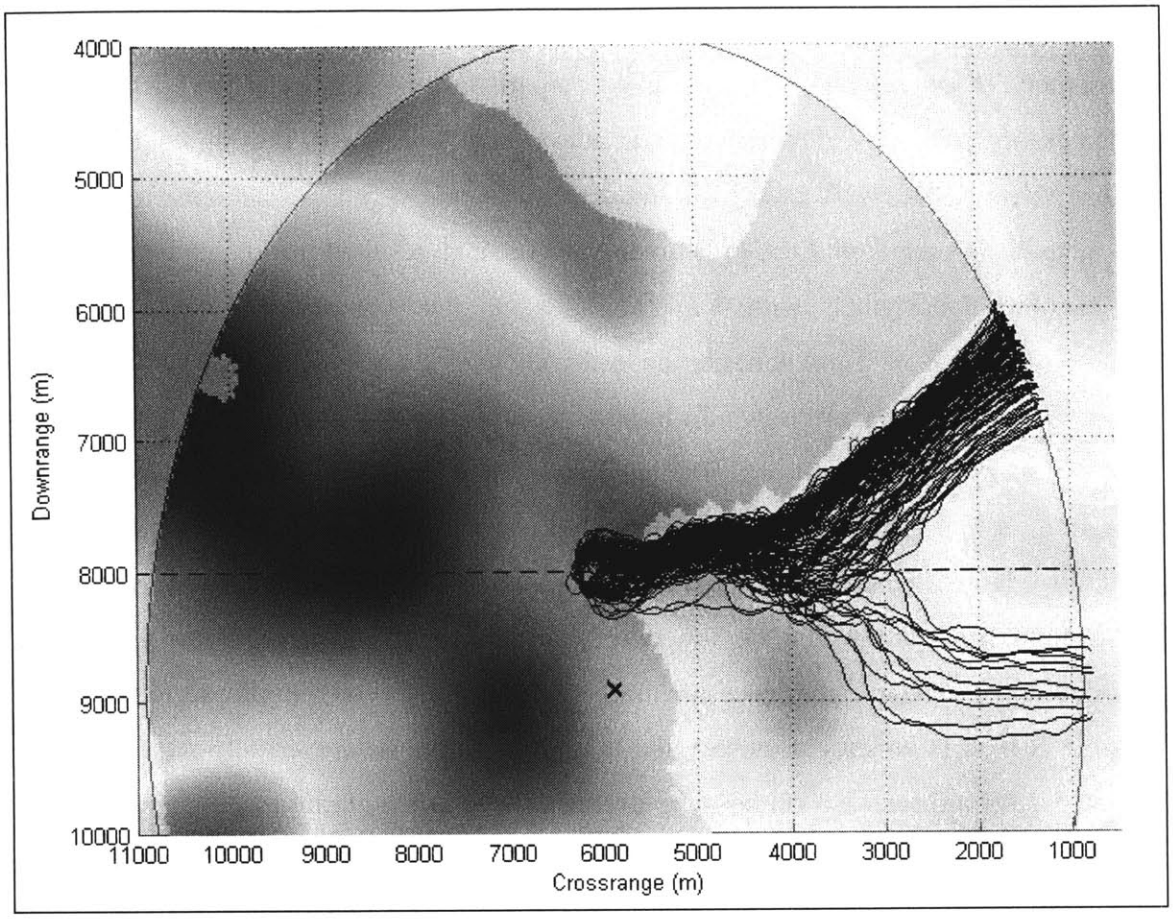

Figure 4-20 Results of 150 TERA evaluations for Scenario #2

First of all, the average run-time per generation for the Matlab implementation of TERA for the results presented in this chapter was 0.238 seconds/generation operating in Windows XP on a 3.19 GHz Pentium 4 processor. Each time the GA within TERA was called it was allowed to run for exactly 600 generations, yielding an average GA run-time of 143 seconds for generating a 40 second trajectory with 2 second instruction lengths.

While the current Matlab implementation is obviously not capable of real-time operation, there exist two major areas in which the run-time of the TERA could be substantially reduced. The first area has to do with the actual coding of the algorithm. Empirical evidence suggests that an increase of *at least* one order of magnitude can be expected when a given Matlab function is coded in a compiled language such as C or Fortran. This estimate agrees with run-time experiments using portions of the TERA code in C++ which suggest at increase in speed of 30-600 times, depending on the amount of looping in the code (Matlab is notoriously slow at for/while loops). This assertion is

further supported in [37], in which a program for auto coding Matlab files into Fortran 90 is presented. Through a series of algorithm comparisons, the authors of [37] demonstrate an increase in speed of 7-1100 times in the automatically generated Fortran code over the original Matlab code, with an average increase of over 330 times. The same algorithms were also hand-coded in Fortran for comparison and resulted in speed increases of 10-1100 times the originals, with an average speed up of 450 times.

If TERA were to experience an order of magnitude decrease in run-time, the current TERA implementation would require only 14.3 seconds to generate a 40 second trajectory using 600 generations. The re-plan horizon of T_R=20 seconds used for the generation of the Chapter 4 results would therefore result in steady-state real-time operability. However, the first time TERA is called after a pop-up threat encounter, the 14.3 second run time would be too excessive for a truly reactive algorithm. If the compiled code were to experience anything over two orders of magnitude reduction in run time (which is entirely possible), the algorithm would be completely real-time. See Section 5.2 for a discussion on possible avenues for reducing initial response time in the case that the compiled code is not sufficiently fast for the first horizon solution. In addition to coding language, optimizing code for run-time can often have substantial effect on run-time, however the current TERA code has not been optimized for speed in any respect.

Furthermore, GAs are inherently highly parallelizable [38, 34]. The production of the new population in each generation and the fitness evaluation can easily be run in parallel, and applications of parallel implementations of genetic algorithms abound in the literature. The current TERA implementation, however, is simply coded in series, leaving yet another potential area for further decrease in computational time in order to support real-time operations.

The second major area for reduction in TERA run-time comes from the algorithm itself. As discussed in Chapter 3, GAs are any-time algorithms; that is, the GA can be stopped at any time during operation and still return a potential solution. This ability to trade off cost for run-time is one of the characteristic of GAs that make them appealing for threat response. Figure 4-21 shows the cost behavior for the first horizon length

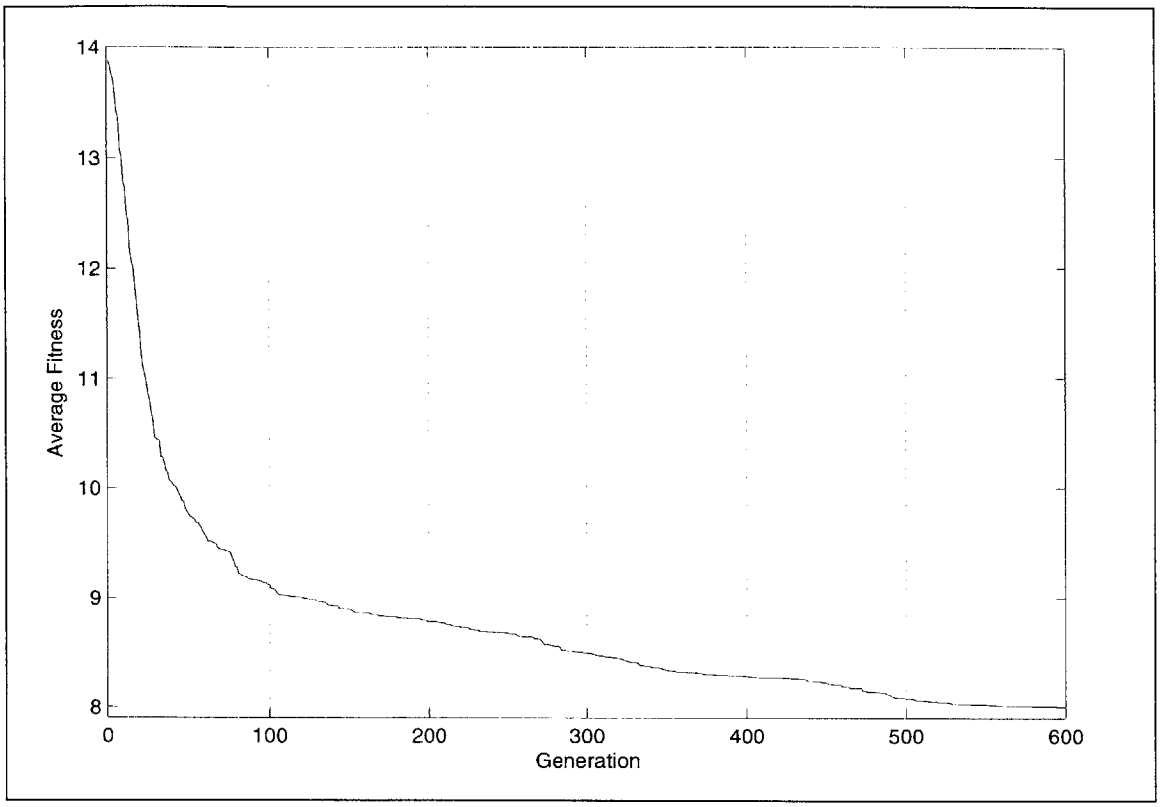

Figure 4-21 Example of cost convergence during Scenario #1

solution for Scenario #1. The data plotted is the average of the minimum cost chromosome from each of the 15 independent TERA evaluations at each generation. The exponential cost behavior is typical of GAs, and demonstrates the kind of trade-off available in terms of penalty incurred for reducing the number of generation available for the solution. For example, a 33 percent reduction in run-time could be achieved if the user were willing to accept an average increase in cost of 4.1 percent in the case shown in Figure 4-21.

Furthermore, the algorithm discussed in Section 3.4 is only a simple GA implementation in terms of the wealth of possible methods for selection mechanisms, mutation operators, and population generation, all of which affect the rate of convergence of the algorithm. By fine-tuning the algorithm structure and parameters, the potential exists for increasing the rate of fitness convergence within the GA. This would result in a direct reduction in the number of generations (and thus run-time) required to achieve a given decrease in cost.

4.2.2 Discussion of results

The simulation results presented have shown that the TERA developed in Chapter 3 is in fact capable of generating intelligent, four dimensional trajectories for threat response. Furthermore, the results, while not deterministic, are clearly repeatable and accurate, which is very important for any potential vehicle guidance algorithm. The response characteristics shown, particularly those in scenarios one and three makes the utility of dynamic trajectory re-planning in response to new threat information abundantly clear. In both cases the trajectories determined by the TERA dramatically decreased the threat exposure time over the predetermined trajectory. In particular these reductions in threat exposure were enabled by the introduction of LOS into the threat cost calculation. While the non-linearity caused by LOS precludes the use of many conventional optimization techniques, the GA easily adapts to the complexity of the ever evolving cost functional, resulting in significant risk reduction through direct LOS manipulation. The ability to explicitly account for LOS to multiple, previously unknown threat location to dynamically re-plan trajectories in low altitude flight makes TERA unique among threat avoidance algorithms in the literature.

Additionally, the results have shown that TERA takes full advantage of the maneuvering capability of the vehicle while exploring the four-dimensional trajectory space in generating evasive trajectories. Even in the fourth scenario, where areas of outstanding reduced risk did not exist, TERA was able to autonomously return intuitive trajectories that evaded the threats limited only by the dynamics of the vehicle.

Chapter 5

Conclusions

This thesis has presented an algorithm for generating four-dimensional aerial vehicle trajectories for the evasion of threats in a military mission environment. The algorithm presented is easily adaptable to new threat situations due to the need for very little *a priori* information about the threat environment. Furthermore, the resultant trajectories are guaranteed to be dynamically feasible through imposing vehicle maneuvering limits during the trajectory generation process.

In this chapter a summary of the work provided in this thesis is provided, followed by a discussion of ideas for future work.

5.1 Summary

In Chapter 2, an overview of a typical low altitude autonomous aerial vehicle mission was presented. In particular, the mission consists of a set of activity points to be visited, which are either reconnaissance or weapons strike points. Some information about threat locations is assumed known in advance of the mission which would be used to route the vehicles around the threats' lethality footprints in order to increase vehicle survivability. While this planning methodology is adequate if the threat knowledge of the environment is perfect, in most operations within enemy lines it is impossible to know

the location of every threat. Because of this, a threat evasive response algorithm is desired in order to re-plan vehicles trajectories in order to increase survivability in the event of pop-up threat encounters. In addition, due to the overwhelming complexity of enumeration of any possible threat encounter for the purpose of contingency planning, it is required that the evasive response algorithm be capable of dynamically re-planning for the vehicle while in flight. It was determined that the evasive response algorithm must be able to generate trajectories that minimize threat exposure risk while taking the vehicle completely out of range of a given pop-up threat. An overview of possible search algorithms from the literature was presented in order to identify the best algorithm to generate the required results, and the class of genetic algorithms was selected for the creation of the threat evasive response algorithm.

In Chapter 3 a background on the operation of genetic algorithms was presented, followed by a detailed description of the development of a genetic algorithm for the purpose of threat evasive trajectory generation. The chromosome representation was established as a set of commanded changes in velocity, heading angle, and flight path angle for the vehicle. The trajectories generated were required to limit commands to the feasible range based on the dynamic capability of the vehicle, resulting in feasible trajectory solutions. A highly customized crossover mechanism was presented that retained the inertial qualities of the trajectories within the candidate population, and furthermore ensured that the joining of trajectories was dynamically feasible. The method of determining the exposure cost of a given trajectory was then discussed, with proper emphasis placed on the whether line-of-sight to each threat exists or not. Line-of-sight was determined to be of significant importance in the minimization of risk from infantry based threats such as shoulder fired anti-aircraft missiles due to the need for visual tracking for the use of these weapons. Finally, the threat evasive response algorithm was defined which makes use of the genetic algorithm developed in a receding horizon fashion in order to take the vehicle completely out of a given threat footprint of arbitrary size.

The threat encounter scenarios simulated were presented in Chapter 4, along with the algorithm results to each threat encounter. Through four different scenarios it was shown that the genetic algorithm based threat evasive response planner developed in this

thesis is quite capable of generating four-dimensional trajectories which take full advantage of the maneuverability of the vehicle in reducing exposure risk to the vehicle. The simulation results demonstrated the algorithm's ability to seek out areas of reduced line-of-sight to the threat, resulting in significant reductions in threat exposure to the vehicle. In addition, evidence was presented suggesting the algorithm is capable of real-time implementation, however no conclusive arguments could be made about run-time due to the Matlab implementation used for the simulation results.

5.2 Future work

This section discusses recommendations for further research into threat evasive trajectory generation.

Algorithm optimization

Unfortunately, genetic algorithms tend to get a bad reputation in terms of run-time. Research proving the real-time feasibility of genetic algorithms for threat evasive trajectory generation would not only benefit in terms of verifying a specific algorithm, but also to increase the viability of genetic algorithms for vehicle path and trajectory generation. This could be accomplished by creating a provably real-time implementation of the algorithm presented in this thesis, which would involve refining the population evolution process (which chromosomes for crossover, which for mutation, whether population overlap should exist [34], etc), and creating a speed-optimized coding in a compiled language. Additionally, speed-up could also be attained by a parallel implementation, which to the author's knowledge would be the first in the literature for genetic algorithm path or trajectory planning.

Reactive response

All of the simulation results presented in Chapter 4 were generated using a horizon length of 40 seconds. Section 4.2.1 mentioned that the TERA implemented in this fashion would be capable of real-time operation given a factor of ten speed-up in solution time (a conservative estimate for a Matlab to compiled code transition). An order of magnitude speed-up, however, may not provide the desired initial reaction time upon

threat encounter. Fortunately, there are several avenues to pursue should a run-time optimized version of TERA not meet the desired reaction time.

The simplest way to accelerate response would be to perform the initial GA solution over a reduced horizon length and number of generations. For example, upon threat encounter the GA could provide a 10 second trajectory using 300 generations in order to provide a reactive response within 1-2 seconds. While this trajectory was being executed a 20 second trajectory could be generated using as many generations as possible within the 10 second window provided by the first trajectory. The TERA could then begin performing in the receding horizon fashion used in Chapter 4. Alternately, a simple reactive heuristic, such as a greedy search, could be employed to provide the initial response spanning the time required to generate the first horizon solution within TERA.

A more involved method for reducing response time would be to employ a set of heuristics to generate solutions to seed the initial GA population with in order to expedite convergence. For example, the initial population could be partially seeded with a set of trajectories that follow straight-line paths radially out from the vehicle position at different heading angles. Some of these solutions would naturally fly away from the threat, providing a reasonable escape route. The heuristic solutions could, however, lead the GA to converge on a local optimum. An interesting research project could include creating a run-time optimized version of TERA and investigating the methods for speeding up the initial threat response in order to complete a real-time threat response guidance package.

Risk modeling

It would be of interest to generate more realistic threat and obstacle risk functions, not only for the algorithm presented in this thesis, but for the evaluation of any potential threat responsive trajectory planning algorithm. For instance, the risk models presented in Section 3.3 assume that MANPAD risk is a linear function of range. This linear drop-off in risk encompasses the fact that as range increase it becomes more difficult for the human operator to visually identify the vehicle. However, when the vehicle is identified it may be correct to assume that the risk no longer is affected by range as long as the vehicle maintains LOS to the threat. It would be of additional benefit to map the threat exposure risk measure used for analyzing chromosome fitness to actual vehicle

survivability in order to compare and verify the effectiveness of differing exposure metrics.

Representation

One of the major assumptions made in the development of the threat evasive response algorithm was that of constant changes in velocity, heading angle, and flight path angle (Section 3.2.1). One way to impose more accurate dynamics on the algorithm could be to use a chromosome representation composed of the maneuvers and trim conditions proposed in [26]. Because each possible trim and maneuver is determined in such a fashion as to guarantee dynamic feasibility, there would be no need for additional checks to maintain feasibility. Furthermore, the maneuver set could be selected to be commensurate with the required resolution and horizon length of the resultant trajectories. That is to say, a coarser subset of maneuvers than those used in the dynamic programming application in [26] could be generated to reduce the computational complexity of the genetic algorithm.

This maneuver based representation, however, would create the need for a new crossover mechanism, due to the fact that the crossover presented in Section 3.2.3 is based on the assumption of constant changes in the control states. If a crossover mechanism that connects an initial position and velocity (six states) to a final position and velocity using maneuvers and trims were created, the maneuver-based representation would be relatively simple to implement.

Multi-objective

In addition to the possibility of generating more realistic threat models, extra objectives could be added to the genetic algorithm for increasing survivability. For example, MANPAD threats are most readily identified by the heat signature created when the first rocket is launched at the vehicle. The algorithm in this thesis took the knowledge of the new threat and generated a trajectory out of the threat's footprint which attempted to reduce the future exposure to that threat, however no attempt was made to try and minimize the risk due to the rocket whose initial launch prompted the detection. While there are several ways the rocket evasion could be approached, one way would be to estimate the time of flight of the missile, and use a fitness function during that time

span that emphasized evading the missile, such as by minimizing altitude and re-orienting the exhaust away from the direction of the missile. In addition, countermeasures such as flares could be incorporated into the response. After the estimated flight time of the rocket had been exceeded, the genetic algorithm fitness function would revert to the original exposure reduction model. The flexibility of the genetic algorithms in cost evaluation makes them particularly well suited for this type of mixed objective problem solving.

Another objective that would be interesting to consider is the addition of weapon firing capability. The fact that the vehicle will often have weapons on board makes it possible that the risk due to a pop-up threat could be reduced by firing in an attempt to destroy the threat. While it may not typically be possible to fire any weapons without human intervention, if the vehicle were to operate in a mission environment where autonomous weapons launch were permitted, the vehicle could potentially greatly reduce risk to itself by using them. The genetic algorithm could be updated to include the decision of whether weapons should be launched, and if so how many and at what times. This would introduce further constraints in that firing weapons would restrict the orientation of the vehicle during target acquisition and launch. Thus, the threat response algorithm could choose between fleeing a specific threat, or turning to fire upon the threat in an attempt to reduce future exposure to the same threat.

References

[1] "RQ-/MQ-1 Predator Unmanned Aerial Vehicle", *Air Force Link Factsheets*, URL: http://www.af.mil/factsheets/factsheet.asp?fsID=122 [sited 13 April 2004], July 2001.

[2] Baker, S., "Predator UAV Marks 50,000 Flight Hours", *AFMC Public Affairs*, AFMC News Service Release 1052, Oct 30 2002, URL: http://www.afmc.wpafb.af.mil/HQ-AFMC/PA/news/archive/2002/oct/1052-02.htm [sited 13 April 2004].

[3] Fulghum, D.A., "On Guard", *Aviation Week & Space Technology*, Vol. 160, No. 7, Pg. 46, Aug 18 2003.

[4] Wall, R., "Precision Kill", *Aviation Week & Space Technology*, Vol. 159, No. 10, Pg. 30, Sept 8 2003.

[5] "Global Hawk", *Air Force Link Factsheets*, URL: http://www.af.mil/factsheets/factsheet.asp?fsID=175 [sited 13 April 2004], April 2003.

[6] Kessner, B.C., "Global Hawk OEF and OIF Images Offer Glimpse of War Contribution", *Defense Daily*, Vol. 219, No. 54, Sept 16 2003.

[7] Peck, T., *Basic Combat Skills/Tactics Flashcards*, 2003.

[8] Fulghum, D.A., "Surviving the Game", *Aviation Week & Space Technology*, Vol. 159, No. 16, Pg. 69, Oct 20 2003.

[9] Schmitt, E., "Iraqis' New Missile Tactics Worry U.S.", *International Herald Tribune*, URL: http://www.iht.com/articles/125476.html [cited May 5, 2004], Jan 19, 2004.

[10] "SA-14 Gremlin", *Federation of American Scientists* (website) URL: http://www.fas.org/man/dod-101/sys/missile/row/sa-14.htm [sited 12 April 2004].

[11] Denton, R.V., Jones, E., Froeberg, P.L., "Demonstration of an Innovative Technique for Terrain Following/Terrain Avoidance – The Dynapath Algorithm", Proceedings from the IEEE National Aerospace and Electronics Conference, May 1985.

[12] Bate, S., Stanley, K., "Heuristic Route Planning: An Application to Fighter Aircraft", Proceedings of the IEEE National Aerospace and Electronics Conference, Vol. 3, Pg 1114-1120, 23-27 May 1988.

[13] Deutsch, O.L., Desai, M., McGee, L.A., "Far-Field Mission Planning for Nap-of-the-Earth Flight", Proceedings of the National Specialists' Meeting, American Helicopter Society, 13-15 Oct. 1987.

[14] Judd, K.B., McLain, T.W., "Spline Based Path Planning for Unmanned Air Vehicles", AIAA GNC Conference and Exhibit, Montreal, CA, August 2001.

[15] Chandler, P.R., Rasmussen, S., Pachter, M., "UAV Cooperative Path Planning", AIAA GNC Conference and Exhibit, August 2000.

[16] Bortoff, S. A., "Path Planning for UAVs", Proceedings of the AIAA American Control Conference, June 2000.

[17] Asseo, S.J., "In-Flight Re-planning of Penetration Routes to Avoid Threat Zones of Circular Shapes", Proceedings from the IEEE National Aerospace and Electronics Conference, 1998.

[18] Norsell, M., "Radar Cross Section Constraints in Flight Path Optimization", 41st AIAA Aerospace Sciences Meeting & Exhibit, January 2003.

[19] Rao, N.S., Phillips, N.L, "Horizontal Plane Trajectory Optimization for Threat Avoidance and Waypoint Rendezvous", Proceedings from the IEEE National Aerospace and Electronics Conference, May 1990.

[20] Vian, J.L., Moore, J.R., "Trajectory Optimization with Risk Minimization for Military Aircraft", Journal of Guidance, Navigation, and Control, Vol. 12, No. 3, May-June 1989.

[21] Asseo, S.J., "Terrain Following/Terrain Avoidance Path Optimization Using the Method of Steepest Descent", Proceedings from the IEEE National Aerospace and Electronics Conference, Pages:1128 - 1136 vol.3, 1988.

[22] Hebert, J., Pachter, M., "Cooperative Control of UAVs", AIAA GNC Conference and Exhibit, August 2001.

[23] Twigg, S., Calise, A., Johnson, E., "On-line Trajectory Optimization for Autonomous Air Vehicles", AIAA Guidance, Navigation, and Control Conference and Exhibit, Austin, TX, August, 2003.

[24] Latombe, J., *Robot Motion Planning*, Kluwer Academic Publishers, Norwell, MA, 1991.

[25] Reif, J.H., "Complexity of the Mover's Problem and Generalizations", Proceedings of the IEEE Symposium on Foundations of Computer Science, pp. 421-427, 1979.

[26] Frazzoli E., "Robust Hybrid Control for Autonomous Vehicle Motion Planning", PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2001.

[27] Kavraki, K., Latombe, J.C., et al, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces", IEEE Transactions on Robotics and Automation, 12(4):566-580, August, 1996.

[28] LaValle, S.M., Kuffner, J.J., "Randomized Kinodynamic Planning",.Proceedings of the 1999 IEEE International Conference on Robotics and Automation, 1999.

[29] Latombe, J., Kindel, R., Hsu, D., Rock, S., "Kinodynamic Motion Planning Amidst Moving Obstacles", Proceedings of the IEEE International Conference on Robotics and Automation, April 2000.

[30] Xiao, J., Michalewicz, Z., Zhang, L., Trojanowski, K., "Adaptive Evolutionary Planner/Navigator for Mobile Robots", IEEE Transactions on Evolutionary Computation, April 1997.

[31] Davidor, Y., "Robot Programming with a Genetic Algorithm", Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering, May 1990.

[32] Dozier, G., McCullough, S., Homaifar, A., Tunstel, E., Moore, L., "Multiobjective Evolutionary Path Planning Via Fuzzy Tournament Selection", IEEE International Conference on Evolutionary Computation, May 1998.

[33] Rathbun, D., Kragelund, S., Pongpunwattana, A., Capozzi, B., "An Evolution Based Path Planning Algorithm for Autonomous Motion of a UAV Through Uncertain Environments", Proceedings of the IEEE Digital Avionics Systems Conference, 2002.

[34] Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Leraning*, Addison-Wesley, Reading, MA, 1989.

[35] Capozzi, B., "Evolution-Based Path Planning and Management for Autonomous Vehicles", PhD thesis, University of Washington, Seattle, WA, 2001.

[36] Fulghum, D.A., "Fire Scout Joins the Army", *Aviation Week & Space Technology*, Vol. 159, No. 16, Pg. 66, Oct 20 2003.

[37] De Rose, L., Padua, D., "Benchmarking FALCON's Matlab-to-Fortran 90 Compiler on an SGI Power Challenge", Proceedings of the IX Brazilian Symposium of Computer Architecture and High Performance Computing - SBAC'97, pp. 285-299, October 1997.

[38] Cristea, V., Godza, G., "Genetic Algorithms and Intrinsic Parallel Characteristics", IEEE Proceedings of the 2000 Congress on Evolutionary Computation, Vol. 1, Pg. 431-436, July 2000.