ONLINE OPTIMAL OBSTACLE AVOIDANCE FOR ROTARY-WING AUTONOMOUS UNMANNED AERIAL VEHICLES

A Thesis
Presented to
The Academic Faculty

by

Keeryun Kang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
August 2012

ONLINE OPTIMAL OBSTACLE AVOIDANCE FOR ROTARY-WING AUTONOMOUS UNMANNED AERIAL VEHICLES

To my lovely wife Gummi,

my beloved children, Jinyoung, Jinwon, and Jinny,

and to my brothers and sisters

ACKNOWLEDGEMENTS

Since the start of the long journey to this dissertation, I have always thought myself very fortunate to have a chance to study at the Georgia Institute of Technology. First of all, I would like to thank Dr. Prasad. He gave me this great opportunity to earn a Ph.D. degree at Georgia Tech. I deeply appreciate his insightful guidance during my research, as well as his continuous support and patience for the last five years. I would also like to thank Dr. Daniel Schrage, Dr. Eric Johnson, Dr. Mark Costello, and Dr. Ho-Il Lee for their comments and suggestions to improve this dissertation.

I am very grateful to Dr. Johnson and his colleagues, Suresh Kannan, Claus Christmann, Dmitry Bershadsky, and John Mooney, for their help in accessing their great UAV system. Without their support, this dissertation would not have been tested in the real world. Especially, I would like to thank Hur Jeong for showing me a model of lifelong enthusiasm and for saving the vehicle from my mistake.

I am grateful to other people who enriched my school life at Georgia Tech; my lab mates shared not only the room but the fatigues and joys of study too: Fahri Ersel Olcer, Gi Yun Chung, Mike Acheson, Ivan Grill, An Binh Vu, and Manuj Dhingra; my Korean friends in the aerospace department and their families, especially Woong-jae's and Jongki's families who shared wonderful times and memories with mine.

I can not show my gratitude enough to my beloved wife and children, brothers and sisters, and parents. They have always been an endless source of encouragement and vitality on this long journey. Without their love, patience and support, I could not have finished this degree. This work is absolutely dedicated to them and to my parents who will be living in my soul forever.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS OR ABBREVIATIONS

$\mathcal{A}$	state real function of flat outputs
a	acceleration vector
$\mathcal{B}$	input real function of flat outputs
B_k	approximated Hessian of cost in BFGS method
$B_{i,}j$	spline basis function
b_l	lower boundary
b_u	upper boundary
$\mathcal{C}$	flat output function of state and input
C_i	Chebyshev polynomial
C_i^j	spline coefficient
c	constraint functions in standard NLP form
$\mathbf{D}$	differentiation matrix
d	slack variable vector
f	nonlinear system equation; nonlinear cost function
G	Collocation matrix from spline coefficient vector to flat output vector
g	nonlinear constraint function; gradient of system cost function
H	Hamiltonian function
H_k	approximate Hessian matrix at k-th iteration
J	cost function; Jacobian of constraint vector
k_i	smoothness of spline
$\mathcal{L}$	Laplacian function
L	integral cost function
L_i	Legendre polynomial at i-th node
l_i	intervals of spline
$\mathcal{M}$	terminal manifold

$\mathcal{M}_q$	merit function
m_i	order of spline
$\mathcal{O}$	space occupied by obstacles
p	p-norm exponent
$p(t)$	three dimensional position of the vehicle; incremental state vector in SQP formulation
p_k	search direction in line search
q	vehicle attitude quaternion
q_k	local quadratic model of nonlinear cost in SQP method
R_d	derivative gain of reference model
R_p	proportional gain of reference model
R_{sensor}	sensor range
r	relative degree of a state
r_s, r_{clr}	safey distance; clearance distance
s	piece-wise linear factor[0,1]
s_i	smoothness of spline
$\mathcal{U}$	admissible input set
U	spline coefficient vector
u	system input vector; logitudinal speed of vehicle
V, V_t	speed; total speed
v	velocity vector; lateral speed of vehicle
w	vertical speed of vehicle
w_i	Gauss weight
x	system state vector; x position of vehicle
y	y position of vehicle; vector in NLP formulation
z	z position of vehicle; flat output
z_{ob}	measured obstacle height

z_{ob}^{f}	filtered height of obstacle surface
α	angular acceleration
α_k	step length in line search
δr	unit grid distance
λ	co-state vector
ν	controllability index; pseudo-control
τ_x, τ_y, τ_z	time constants
ω	vehicle attitude rate
ξ	defect of cost; flat output vector
ψ_0	initial constraint function
ψ_f	terminal constraint function
ψ_t	path constraint function
∇	gradient
∇_x	gradient by x
∇^2	Hessian of a function
$\lVert \cdot \rVert$	Euclidean norm
$\dot{}$	first derivative
$\ddot{}$	second derivative
${}^{(r)}$	r-th derivative
^{T}	vector or matrix transpose
*	optimal value
ACL	Autonomous Control Level
ADLAT	Advanced Low Altititue Technique
ANN	Adaptive Neural Network
BFGS	Broyden-Fletcher-Goldfarb-Shanno
B-spline	Bezier-spline
CGL	Chebyshev-Gauss-Labatto

CLF	Control Lyapunov Function
CONOPT	A solver for large-scale nonlinear optimization developed by ARKI Consulting and Development A/S in Bagsvaerd, Denmark.
CTG	Cost-To-Go
DDP	Differential Dynamic Programming
DIDO	A pseudo-spectral based optimization tool
DOF	Degree of Freedom
DP	Dynamic Programming
GPOPS	Open-Source General Pseudo-spectral Optimal Control Software
GUST	Georgia Tech UAV Simulation Tool
INTOPTOA	Integrated Optimal Obstacle Avoidance
IPM	Interior Point Method
IPOPT	Interior Point Optimizer
LG	Legandre-Gauss
LGL	Legandre-Gauss-Labatto
LIDAR	Light Detection And Ranging, also LADAR
MA	Manuever Automaton
MILP	Mixed Integer Linear Programming
MINOS	A software package for solving large-scale optimization problems
MOUT	Military Operations in Urban Terrain
MPA	Motion Premitive automation
MPC	Model Predictive Control
NLP	Nonlinear Programming
NOE	Nap-Of-the-Earth
NP-Hard	Non-deterministic Polynomial-time Hard
NPSOL	Nonlinear Programming Solver
NTG	Nonlinear Trajectory Generation

OCP	Optimal Control Problem
PCH	Pseudo Control Hedging
PD	Proportion-Derivative
PFN	Potential Field Navigation
PID	Proportion-Integral-Derivative
PS	Pseuso-Spectral
PSOPT	Pseuso-Spectral Optimizer
QP	Quadratic Programming
RH	Receding Horizon
RHC	Receding Horizon Control
RIOT	Radical Image Optimization Tool
RPV	Remote Piloted Vehicle
RRT	Rapid Random Tree
SNOPT	Sparse Nonlinear Optimizer
SQOPT	Sequential Quadratic Optimizer
SQP	Sequential Quadratic Programming
TA	Terrain Avoidance
TF	Terrain Following
TFR	Terrain Following Radar
TPBVP	Two Point Boundary Value Problem
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle

SUMMARY

This thesis presents an integrated framework for online obstacle avoidance of rotary-wing unmanned aerial vehicles (UAVs), which can provide UAVs an obstacle field navigation capability in a partially or completely unknown obstacle-rich environment. The framework is composed of a LIDAR interface, a local obstacle grid generation, a receding horizon (RH) trajectory optimizer, a global shortest path search algorithm, and a climb rate limit detection logic.

The key feature of the framework is the use of an optimization-based trajectory generation in which the obstacle avoidance problem is formulated as a nonlinear trajectory optimization problem with state and input constraints over the finite range of the sensor. This local trajectory optimization is combined with a global path search algorithm which provides a useful initial guess to the nonlinear optimization solver. Optimization is the natural process of finding the best trajectory that is dynamically feasible, safe within the vehicle's flight envelope, and collision-free at the same time. The optimal trajectory is continuously updated in real time by the numerical optimization solver, Nonlinear Trajectory Generation (NTG), which is a direct solver based on the spline approximation of trajectory for dynamically flat systems. In fact, the overall approach of this thesis to finding the optimal trajectory is similar to the model predictive control (MPC) or the receding horizon control (RHC), except that this thesis followed a two-layer design; thus, the optimal solution works as a guidance command to be followed by the controller of the vehicle.

The framework is implemented in a real-time simulation environment, the Georgia Tech UAV Simulation Tool (GUST), and integrated in the onboard software of the rotary-wing UAV test-bed at Georgia Tech. Initially, the 2D vertical avoidance capability of real obstacles was tested in flight. The flight test evaluations were extended to the benchmark tests for 3D avoidance capability over the virtual obstacles, and finally it was demonstrated on real obstacles located at the McKenna MOUT site in Fort Benning, Georgia. Simulations and flight test evaluations demonstrate the feasibility of the developed framework for UAV applications involving low-altitude flight in an urban area.

CHAPTER I

INTRODUCTION

1.1 UAVs and Evolution of UAV Autonomy

The development of unmanned aerial vehicles (UAVs) started from an interest in replacing manned aircraft in dangerous or dull tasks in military operations, and UAVs quickly emerged as efficient and cost-effective aerial platforms for tactical reconnaissance and surveillance, as their operation became practical. Currently, different forms of UAVs are operating or under development as seen in Figure 1, mainly for military purposes in surveillance, reconnaissance, targeting, and even in the near future for combat operations.

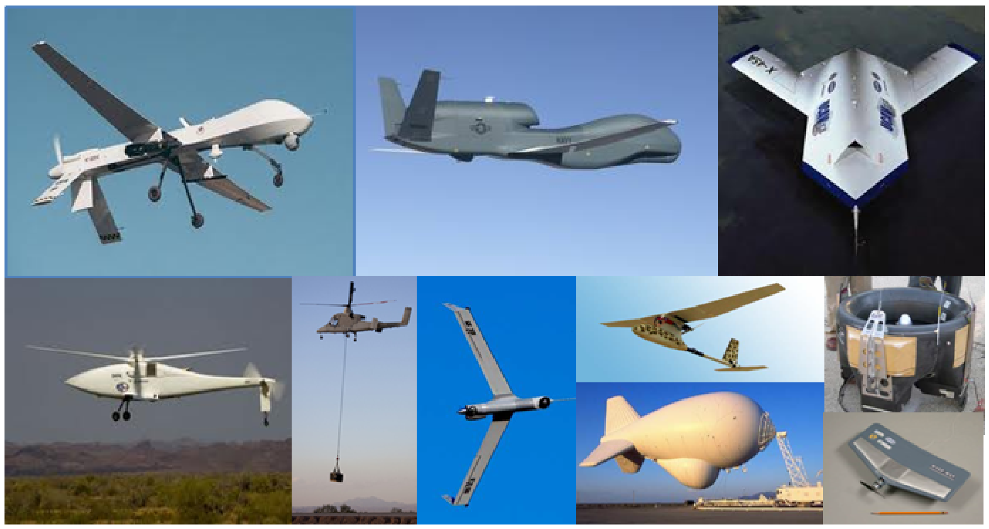


Figure 1: Various forms of UAVs are operating or being developed.

Obviously, the capability and versatility of UAVs have contributed to recent growing attention toward them: UAVs are the best-suited aerial platforms to replace manned aircraft in 3-D missions that are dull, dirty and dangerous. The political and human cost is lower if the mission fails, and the probability of mission success is higher than for manned aircraft [1]. In addition, UAVs are cost-effective to procure and to maintain [10], mainly because they lack the requirement of considering human factors for their operation. Today they are being employed in roles and applications that their designers never envisioned as their unanticipated flexibility and mission capability have been proven in recent conflicts and crises. This proven versatility of UAVs has led to the consensus that UAVs are indispensable assets that play key roles in modern technology-intensive battlefields, even taking over some conventional roles usually filled by manned aircraft. Therefore, the development of UAVs has been spreading rapidly worldwide for various military and civilian purposes.

UAVs have also been called unmanned aircraft systems (UAS), remotely piloted vehicles (RPV), drones, or robot planes, but the name UAV is widely accepted among associated communities. A UAV is described as either a single vehicle or a system, usually consisting of three to six air vehicles, a ground control station, and support equipment [10]. According to the definition from the US Department of Defense (DoD), a UAV is [1]:

A powered, aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable, and can carry a lethal or non-lethal payload. Ballistic or semi ballistic vehicles, cruise missiles, and artillery projectiles are not considered unmanned aerial vehicles.

Along with the above definition, another distinguishing feature of UAVs is the ability to execute missions with either minimal or no dependence on outside instructions; this is known as UAV *autonomy* or *built-in intelligence* [22]. Numerous technologies have been developed for the subsystems, systems, or system of systems level of UAV autonomy, and those technologies range from low-level remote piloting to the highly advanced multi-agent collaboration. So, there have been efforts to establish a common standard to measure the level of complexity and capability of UAV autonomy. One

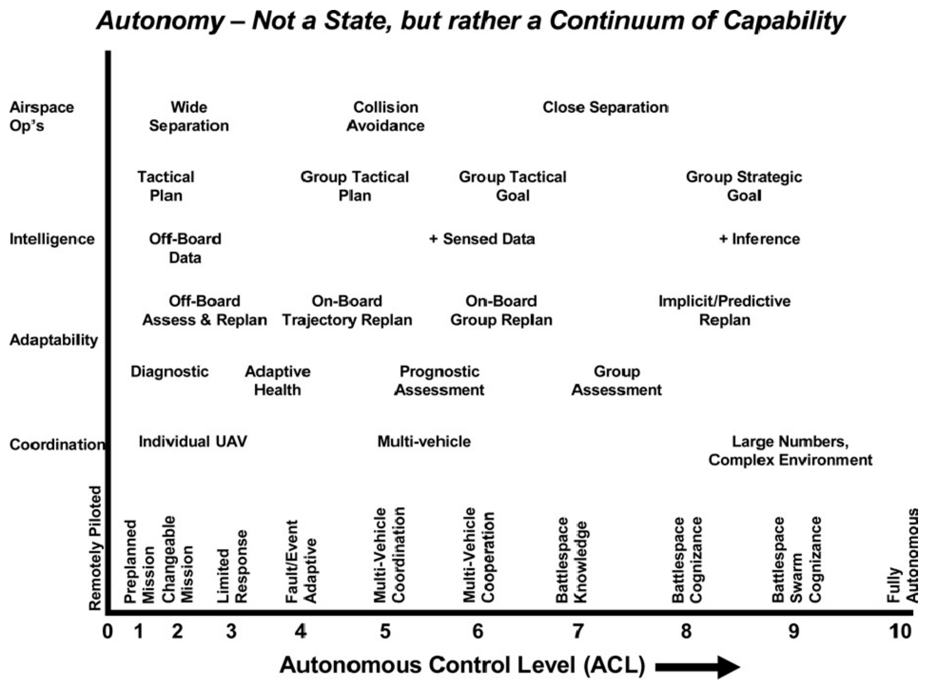


Figure 2: Machine decision capabilities versus autonomous control level [22]

standardized metric is the autonomous control level (ACL) [1, 22, 120] and Figure 2 depicts the ACLs and the corresponding capabilities.

In addition to ACLs, the autonomous technologies for a single UAV can be classified into three hierarchical layers [11, 102]: flight control, vehicle management, and mission management. The first layer includes automatic flight control technologies or functions to stabilize the vehicle and guide it along the command path. This layer includes vehicle stabilization, autonomous take-off and landing, and adaptive trajectory tracking. In particular, for UAVs, there has been a trend of applying current state-of-the-art control techniques rather than using classical PID controls among UAV research groups because the controller is required to have high performance to overcome the challenges inherent in the new applications of UAVs. The main feature of UAVs, that they are uninhabited by human pilots, brings its own set of pros and cons to controller design. On the one hand, the controller design need not be concerned with human limitations and safety; on the other hand, the controller must be highly robust and flexible to make up for the lack of human intelligence in abnormal decision-making situations. Damage or fault tolerant control [33], model predictive control [74], and nonlinear adaptive control based on neural network (NN) [62] are being actively studied and applied to the UAV control systems. The second layer of UAV autonomy, the vehicle management layer, includes the technologies associated with coordination and navigation in mission environments. Command generation to the control layer, vehicle configuration management, operational mode management, fault detection and identification, multi-sensor fusion for navigation, and obstacle avoidance functions can be sorted in this layer. Finally, the mission management layer includes the technologies dealing with mission management, such as generation of mission trajectories, or coordination between multiple UAVs. The time-line management of each vehicle, mission execution sequences, in-flight mission reassignment, and formation flight management may be included in this layer.

Recent realization of high-level autonomy functions and real battlefield-proven versatility can be attributed to revolutionary breakthroughs in computer technologies. Equipped with more powerful microcomputers, accurate micro navigation devices, high-precision surveillance sensors, wide range communication systems, and high speed data-links, future UAV systems are expected to be outfitted with high level TRL (Technology Readiness Level) technologies. The anticipated high-level technologies in the near future have prompted research on the possibility for expanding the UAV's operation areas beyond military to public and possibly commercial applications [26]. These new application areas include search and rescue [48], surveillance on sites hit by natural disasters (i.e., tornado, earthquake or flood), typically inaccessible or dangerous to humans, or sites where high radioactive radiation is present [125], as depicted in Figure 3, which shows the Fukushima nuclear power plant after the melt-down caused by the earthquake and tsunami in Japan in 2011.

Also, severe weather observation [29], traffic monitoring [109], urban surveillance

Figure 3: After the melt-down of the nuclear power plant at Fukushima, Japan in 2011

[117], indoor navigation and surveillance [121], and even filming for movies or advertisements are being forecasted as new application areas for UAVs. Obviously, some of the new applications inevitably require the near-terrain flight capability over urban areas. For such missions, rotary-wing UAVs have been considered the best-suited because of their unique vertical and horizontal flight capabilities such as hovering, vertical climb and descent, level flight with any heading angle, and even backward flight. Therefore, new emerging UAV applications requiring the low-altitude flying capability have motivated ongoing research on autonomous obstacle avoidance in obstacle-rich and unknown environments, or on collision avoidance in airspace filled with various aircraft swarms. Indeed, this thesis is one of such research efforts.

1.2 Autonomous Trajectory Planning

The term *planning* has been commonly used in robotics to roughly define the problem of how to move, and *trajectory planning*, otherwise referred to as trajectory generation, path planning, or more often motion planning in robotics, is broadly defined in control literature as the construction of inputs to a nonlinear dynamical system that drives it from an initial state to a specified goal state [83]. Otherwise, we may describe trajectory planning as a computation to create a desired trajectory (or path), which is simply the time series of position, velocity, and acceleration, accounting for the vehicle's kinematic and dynamic properties to accomplish the desired flight pattern of a mission.

1.2.1 Hierarchy of Planning

The majority of trajectory planning for UAVs has been done with hierarchical layers [11], similar to the layers of autonomous technologies, shown in Figure 4, which depicts the top-down connections of mission planning, trajectory planning, and vehicle control.

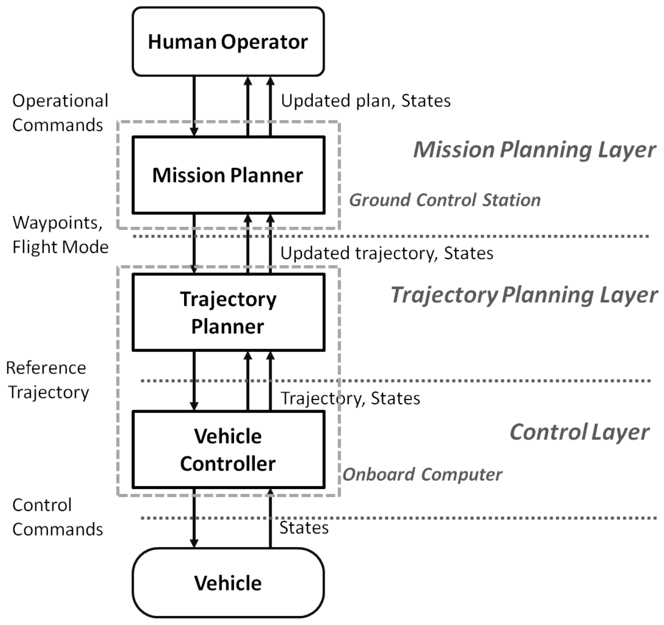


Figure 4: Hierarchical layers of planning and control of UAV

The mission planning layer is for the high level human-UAV inter-operations. Decisions and inputs of human operators, flight or mission plans, and communications between operators and the vehicles are generally handled and processed on separate computational systems in the ground control station. Mission monitoring, mission assignments, task scheduling, and generation of a global path are typically executed in this layer of planning. Waypoints of the mission, task assignments, and a global mission path are the typical outputs of the mission planning. Global trajectory optimization using vast sources of tactical data and computational resources is usually involved in path planning in this layer [12]. The resultant waypoints, tasks, or coarse paths are transferred to the lower layer, the trajectory planning layer, to generate dynamically tractable or feasible paths for the vehicle. The trajectory planning layer is for the guidance of the vehicle from its present state to its desired path or to the waypoints determined by the mission planner. In general, the trajectory planner generates local reference trajectories to be followed by the controller. One typical trajectory planner most UAV systems are adopting is the waypoint navigator that generates the guidance command either to pass by or arrive at the waypoint by the sequence of the entire waypoint set. Finally, the control layer has the role of vehicle stabilization and command trajectory tracking.

Currently, the majority of UAVs are still being flown by remote human piloting or by the automatic tracking of waypoint-based plans predetermined on the ground. In fact, these operational schemes still largely need human operation during ordinary flight. However, the future UAV systems are expected to have lower human dependency even in environments like an urban operation of UAV swarm. Obviously, the automation of trajectory planning as well as improving current online trajectory planning are crucial to the mission capabilities of future UAV systems, and automatic obstacle avoidance in uncertain environments is one of the typical aspects of trajectory planning automation that should be realized with priority.

1.2.2 Obstacle Avoidance

In robotics, obstacle avoidance is known as a representative task of trajectory planning to overcome obstacles during movement to a destination. One mathematical description of obstacle avoidance can be given as: *There is obstacle avoidance for any vehicle, if $p(t) \notin \mathcal{O}$, where $p(t) \in \mathbb{R}^3$ is the position of vehicle and $\mathcal{O} \subset \mathbb{R}^3$*

represents the set of all obstacle spaces [114], or we may describe it informally in the manner of control literature as *any control efforts to make the plant's position satisfy the non-intersecting position constraints to obstacles* [83]. The above descriptions of obstacle avoidance imply that it is a typical trajectory planning problem to guide the vehicle to find a safe path through free space in the environment with respect to the obstacle geometry, the vehicle dynamics, and other constraints. Obstacle avoidance has been a major subject in robotics in the development of articulated robot manipulators [43, 71], unmanned ground vehicles (UGVs) [51, 79, 119, 123], or even robot soccer [49].

In fact, it is a recent trend that the problem of obstacle avoidance is treated as a trajectory planning problem, which implies a guidance problem to find the local trajectory being followed, but in much of the literature, it has been characterized as a problem of mission planning to find the global path for military aircraft at the operational or strategic level, i.e., the path that maximizes safety and survivability and minimizes vulnerability to expected threats, which is solved by deliberate optimization or even by stochastic game theory [12, 104]. A great variety of methods using either heuristics or optimizations have been invented for robotics and aerospace applications. Detailed reviews on such methods will be addressed in later sections.

1.2.3 Terrain Following and Nap of the Earth

Terrain following (TF), otherwise known as contour flight, terrain avoidance (TA), and the nap of the earth (NOE) are the representative low-altitude tactical maneuvers of military aircraft. TF has more than a half century of development, specifically for the deep penetration attack aircraft [13, 66] and missiles [86, 136], to provide effective guidance to maintain minimum clearance over terrain and to fly fast, while minimizing the probability of being detected by enemy surveillance either on the ground or in the air [99]. Unlike TF, TA is a horizontal guidance problem of flying around hills and avoiding threats, and NOE is a three-dimensional guidance problem that deals with much lower vertical and lateral trajectories [19, 55]. In fact, NOE has been known as the specific tactical maneuver of helicopters [20, 21, 25] because the helicopter's high maneuverability at low speed is best suited to such a maneuver. On the other hand, fixed-wing aircraft have difficulty in maneuvering at low speed due to the limitation of the stall. During NOE flight, helicopters tend to fly within a few feet of power lines, trees, hills, or other ground objects. Figure 5 shows the different low-altitude maneuvers of a helicopter.

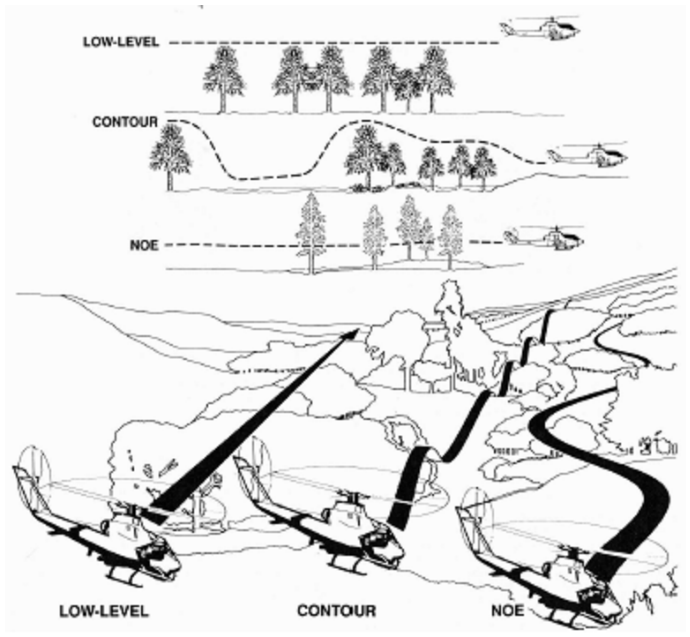


Figure 5: Three typical low-altitude maneuvers of rotor-craft [19]

TF has been explored as either an offline trajectory optimization problem [41, 73, 88, 108, 129] or a guidance and control problem to steer the vehicle in vertical plane using the terrain contour measured by the forward looking terrain-following radar (TFR) and radio altimeter (RA). One renowned non-optimization based TF algorithm is the Advanced Low Altitude Technique (ADLAT), which is still being used in many military aircraft [13, 66, 72, 77]. ADLAT generates the flight path angle command by computing constant vertical acceleration parabolas for current flight conditions and dominant (highest) terrain point within TFR range [40]; in other words, ADLAT builds the TF trajectory composed of segments of constant acceleration pull-up and push-over or reverse combination.

From the early years since the TF concept was invented, many researchers have concentrated on trajectory optimization techniques to find the minimum deviation trajectory above terrain [50, 134, 136]. A survey of optimization-based TF algorithms reveals that, in order to resolve the issues of numerical optimization and limited computing power, some past optimization-based methods solved a finite horizon optimal control problem formulated as linear quadratic (LP) programming, and the optimal control was tracked by the flight controller [41, 72, 77]. This approach is similar to current MPCs except that MPC usually seeks the direct control command to the vehicle. Harpern [50] presents good reviews of the early optimization-based TF/TA methods.

Like other trajectory planning problems, TF problems have also been explored by a variety of methods from the past: combinational searching methods using dynamic programming (DP) [7] or A^* algorithm [53]; the branch and bound method in conjunction with the DP [131]; spline based optimization methods [41]; optimal control based methods [108]; direct optimization methods using direct collocation [52, 100]; even genetic algorithms (GA) and neural networks have been applied to TF problems [50]. Similar to the recent trend of trajectory planning which will be presented later, the survey discovered that practical implementation of TF algorithms in actual aircraft systems [66, 72] uses a search algorithm like A^* to get a mission-level coarse optimal path, and it uses a guidance system like ADLAT or linear optimal control to generate a local path over the terrain. The computed local path is used either for the automatic TF guidance or for the provision of guidance information to pilot via avionics such as a heads up display (HUD) or helmet mounted display (HMD).

For some time, the global TF paths have been determined by optimization methods using the calculus of variation or *indirect* optimal control methods [88, 108, 129]. However, the optimal control based methods generally require solving the two-point boundary value problem (TPBVP), which is computationally expensive, and often the solutions are difficult to find even though the method can provide the global TF path of which optimality is theoretically guaranteed. The diversity of optimization methods for TF is similar to those of trajectory planning, so they will be presented in a later section about the optimization-based trajectory planning methods.

The trajectory planning for NOE flight is similar to that of TF or obstacle avoidance problems except that the NOE considers a three-dimensional path more proximate to the terrain and obstacles to maximize cover and concealment. This distinction results in a significant difference in formulation between the NOE problem and the obstacle avoidance problem. The NOE problem adopts optimization to minimize the deviation from terrain and obstacles with the performance index having clearance as a term, whereas the general obstacle avoidance problem usually takes the clearance as a constraint, as its definition implies. Therefore, roughly speaking, NOE flight might be the extension of TF to three-dimensional space.

NOE path planning inherently assumes incomplete information about the flight region. Although the large scale terrain features and structures might be completely surveyed a priori, exact information about small obstacles like trees, small buildings, power lines and wires, etc., is not easy to acquire and also highly variant with time. In order to tackle this aspect of trajectory planning for NOE, Cheng [19, 20] proposed a three-stage of planning, long-term mission planning for a template path using large scale known terrain geometry and obstacles to set up the waypoints, mid-field trajectory planning with a local map and known obstacle information to get the trajectory between two waypoints, and finally near-field in-flight trajectory planning using onboard obstacle sensors to generate the safe local avoidance trajectory while minimizing the deviation of the mid-term path. This local avoidance trajectory generation for NOE flight is the main subject of this thesis.

1.3 Survey of Trajectory Planning Methods

Trajectory planning or motion planning has been an important research topic in the artificial intelligence community in the field of robotics and aerospace for decades, and numerous methods have been developed for a variety of applications in both areas. A recent book by LaValle [83] covers a broad range of motion planning methods, and other excellent survey works by Schwartz [116], Hwang and Ahuja [56], and Latombe [81] have been used as references for early motion planning methods. In addition to early surveys, recent works [47, 96] provide good overviews of trajectory planning methods in the UAV field.

Trajectory planning methods can be classified in different ways: by the scope of planning (deliberate or reactive, otherwise, global or local planning); whether it considers differential constraints (i.e., vehicle dynamics) or not; whether it uses optimization technique or not; whether it uses heuristics or not; the probabilistic or not, etc. As the main interest of this thesis is trajectory planning for obstacle avoidance, those methods that have been widely applied to obstacle avoidance for UAVs are selectively surveyed.

1.3.1 Methods without Differential Constraints

The planning methods in this category do not formalize any differential constraints (kinematics or dynamics) in problem formulation and purely focus on finding a safe path in configuration space. Thus the planning is relatively easy compared to the methods that include vehicle dynamics, but the resultant path might be dynamically infeasible. However, despite this disadvantage, the computational advantages support using these methods as the base trajectory generator for other methods that account for the vehicle dynamics. There are three well known major sub-classes in this category: cell decomposition methods, roadmap methods, and potential field methods.

The cell decomposition methods rely on partitioning of the configuration space into a finite number of cells in each of which collision-free paths can be easily found. In this method, the configuration space is the set of all possible configurations of the vehicle, usually referred to as the vehicle state space, and its dimension corresponds to the degrees of freedom of vehicle motion [83]. The planning problem in the cell decomposition method is the problem of finding a sequence of neighboring cells including the initial and final destination. Depending on the detailed decomposition, i.e., how to make free space into smaller convex polygons, there are different variations: Rectangular cell decomposition [36], quad-tree or octree decomposition [5], trapezoidal decomposition, cylindrical algebraic decomposition, connected ball [130], etc. For searching neighboring cells, heuristic search methods such as A^* [53] and D^* [122] have been commonly used.

Roadmap methods convert the path planning problem into the problem of constructing a network of segmented paths that spans the obstacle-free space where the problem is to find the best sequence of connecting path segments from the initial position to the goal by search algorithms such as A^*, D^*, and dynamic programming. Visibility graph methods [32, 101] and Voronoi diagram methods [54, 85, 128] are representative; the freeway and the silhouette method [16] also can be sorted in this class.

The above two classes of methods are *complete*, which means in planning theory that they can find the solution if it exists; and they are computationally efficient, so they can compute the entire path fast. However, the resultant path always touches the marginal boundaries of obstacles, which is sometimes unnecessary.

Potential field methods, otherwise known as potential field navigation (PFN), were first proposed by Khatib [71], who applied the classical potential flow equation in fluid dynamics to the path planning of mobile manipulators and mobile robots [76, 119]. In PFN, obstacles are represented as repulsive potential functions, the sources in fluid dynamics, whereas the destination point is represented as an attractive potential function, or the sink. The problem formulation corresponds to the procedure of constructing and placing repulsive and attractive potential functions depending on the obstacle configuration; the potential function is partially differentiated to come up with the force to control the vehicle and to find the path. The numerical integration of the partial differential equation usually results in smooth and dynamically tractable paths around the obstacles to the destination. However, it is widely known that PFN has some disadvantages: The classical PFN is *incomplete*, for the vehicle might get trapped in local minima depending on the obstacle configuration (if the shape is oval)[47], so, the complete path to the destination may not be obtained even though it exists. Obstacles are modeled as soft constraints with continuous and differentiable potential functions. They cannot be modeled as hard constraints because of the difficulty of the potential functions to describe the obstacle shape exactly; thus the hard avoidance is not possible in PFN and the vehicle may approach too closely to obstacles, unnecessarily breaking the desired clearance. There have been efforts to fix these disadvantages by modifying the classical PFN approach, and the use of harmonic potential functions [23] is one of such efforts.

1.3.2 Randomized and Probabilistic Methods

Randomized methods were invented to circumvent the computational complexity of global path planning and to increase the completeness of the algorithm. The probabilistic roadmap (PRM) method, the rapid random tree (RRT) method, and the motion primitive automation (MPA) methods are sorted in this category. The PRM algorithm was first introduced by Kavraki et al. [69]. PRM algorithms combine an offline construction of a roadmap and randomized online selection of a path from the roadmap. Online selection of the path is done by computing the shortest path from the current position to the closest node point on the roadmap. The completeness of PRM has been proven in a probabilistic sense [38]; however, this algorithm cannot encompass the vehicle dynamics and the rapidly changing environment with moving obstacles due to the offline computation of the roadmap.

LaValle [82] introduced the rapidly-exploring random tree (RRT) method as an alternative randomized method that can produce dynamically tractable trajectories with hard avoidance guarantees. RRT finds the path by building the tree of tractable segment paths and propagating branches toward randomly generated intermediate target points until at least one branch reaches the destination. A significant feature of RRT is that the resultant trajectories are highly tractable by the vehicle and, under appropriate conditions, the RRT algorithm has been proven probabilistically *complete*, so it can find the path to the destination if it exists [39].

However, it has been noted that the above two methods have difficulty in including more accurate vehicle dynamics or vehicle motions. To resolve this issue, Frazzoli [38] introduced a randomized method using motion primitives of the vehicle, the so-called maneuver automaton (MA), which represents the predefined set of trim and dynamically feasible maneuvers [38]. In MA methods, RRT uses the MAs to expand the path in the obstacle field, and the path is selected by the optimal selection by dynamic programming that minimizes the value function.

1.3.3 Reactive and Deliberate Planning

The terms *reactive* and *deliberate* can be otherwise referred to as *local* and *global* respectively, representing the scope of planning. Reactive planning methods use only the local knowledge of an obstacle field usually obtained from the sensors, such as a visual camera [112] or a scanning laser [44], and the method continuously updates the local trajectory *patches* as time advances. So, the reactive method can handle the uncertainties in the obstacle field, such as obstacles unknown a priori or moving obstacles. However, because it only takes the local area into account and not the entire obstacle field, it may not find a complete trajectory to the destination, let alone an optimal one. Therefore, there has been a trend of combining reactive planning with the global planning method to "adjust" the portion of global path to the changes in the obstacle field; the computation time should be sufficiently fast to deal with sudden changes in avoidance situations, so many reactive approaches have used heuristic trajectory planning methods. The work of Redding et al. [107] is a typical example that used the combination of RRT and Dijkstra's algorithm [31] and Hwangbo et al. [57] used RRT and A^*.

On the other hand, deliberate planning is mostly related to military mission planning systems [12], and it deals with the global trajectory, taking into account broad mission elements, such as other friendly or enemy vehicles involved in the mission, available fuel, armaments, threats on the ground and in the air, etc. For obstacle avoidance, deliberate planning uses all known information about obstacles and the path generation method that can produce a complete path. In fact, any complete planning methods introduced in the previous section can be used in the deliberate planning, but the optimization-based global path planning has been used most often in deliberate planning because deliberate planning seeks the best path that can satisfy the mission objectives such as the maximum mission effective path, the maximum survivability path, or the minimum fuel-consumption path.

1.3.4 Optimization-Based Methods

Trajectory planning has been one major area where many mathematical and numerical optimization techniques are actively applied to get the optimal path satisfying the constraints given in the problem specification and the criteria such as shortest path, minimum time, minimum energy, minimum fuel, etc. The trajectory planning by optimization is often referred to as *trajectory optimization*.

Trajectory optimization has the advantage that the resultant path is backed by the optimality from the background theory of optimization, and it intrinsically includes the computation and selection process for determining the best path from the set of feasible candidates satisfying the requirements of the problems other heuristic methods cannot provide. However, it may need complicated problem formulations and significant computational resources, which actually have prevented trajectory optimization being used for online applications in the past. Specifically, for the formulation of trajectory optimization, formalizing complex obstacle geometry, full vehicle dynamics, and associated nonlinear constraints into mathematical forms are not easy tasks, and as the complexity of the optimization increases, the resultant optimization problem usually ends up NP-hard [61]. Even though the initial guess is provided, the convergence time might increase significantly [61, 135]. However, despite those disadvantages, by the use of approximations to reduce the computational complexity, such as point mass vehicle assumption, simplified obstacle geometry, linearized complex constraints, etc., trajectory optimization has been used widely for offline global path planning in particular.

The following strategies have been applied to obstacle avoidance. The time-optimal obstacle avoidance with or without state and input constraints [126, 127] has been a common policy for some time, and even recently, the time-optimal approach incorporating the vehicle limit parameters into the obstacle avoidance was introduced by Moon and Prasad [94], which is the basis of this thesis. Regarding safety considerations, much of the literature has taken the clearance distance as a hard constraint, but there are other optimal approaches based on MPC formulating the clearance as a soft constraint embedded in the cost function [118]. Trajectory optimizations have been applied to the terrain following problems too, having both time and vertical clearance in their cost function [88, 129]. Therefore, the role of dealing with safety in trajectory optimization depends on the problem policy: *Is safety a soft constraint or a hard one?*

A classical discrete optimization technique being widely used for searching the global optimal path is dynamic programming (DP), which is based on Bellman's *principle of optimality*, simply stated: *if a trajectory is optimal, the end-portion of it should also be optimal* [7]. DP has been widely used to search for the optimal path in cell-decomposition methods and graph search methods, mostly in order to build the optimal path considering the weighted cost of each segmented path without considering any dynamics or constraints. DP is inherently unable to take into account the dynamics and the constraints in its original formulation, so optimization using DP for dynamics problems should discretize the problem till the system dynamics is inconsequential. This approach usually ends up with huge problem dimensions. For dynamic optimization problems, the extension of DP to the differential dynamics systems, referred to as differential dynamic programming (DDP) [59], has been used in various trajectory optimization problems of dynamical systems [80] even with constraints [84].

Other optimization methods that can handle nonlinear dynamics can be categorized into two major groups, *indirect* methods and *direct* methods. Betts [9], Conway [24], and Stryk [124] provide a thorough review of the major methods of both categories. Indirect methods are based on the optimal control theory rooted in the calculus of variation and the necessary conditions of optimality derived from Pontryagin's Maximum Principle [103]. The simultaneous differential equations of states and co-states are analytically derived from the necessary conditions of optimality and the transversality conditions, respectively. These equations form the so-called two point boundary problem (TPBVP) [14, 15], which is solved by numerical solution techniques, such as gradient descent, shooting, and collocation method [9]. It is well known that indirect methods can provide more accurate (local or global) optimal solutions but using indirect methods for trajectory optimization has several problems associated with TPBVP. For one thing, it requires daunting analytical derivation of differential equations, which becomes a challenging task if the problem includes complex nonlinear system dynamics or constraints. In addition, the convergence of the solution is sensitive to initial guesses. Also, it is difficult to build initial guesses of co-states, which usually do not have a physical meaning. Along with the above disadvantages, indirect methods have less flexibility for online applications, and for this reason they have been often limited to offline trajectory optimization. Kim [73] and Menon [88] show typical examples of the use of indirect methods of trajectory optimization.

The other group of optimization methods is direct methods. Direct methods convert the optimal control problem (OCP) into nonlinear programming (NLP) using discretization, or collocation, or parametric approximation of states and inputs, and then they solve the converted NLP directly with numerical NLP solvers. Direct shooting [9], direct collocation nonlinear programming [30], pseudo-spectral methods [42], and the spline-based method [91] are representative direct methods that have been intensively studied for many trajectory optimization problems. This thesis surveyed the pseudo-spectral method and the spline-based method as optimization solvers and finally chose the spline-based method. Details of both methods will be described later. It is known that direct methods are more robust and flexible because of no need to derive necessary conditions and, thus, no need for initial guesses for the co-states [9]. However, depending on the level and accuracy of discretization or parameterization, the dimension of optimization can be varied significantly, affecting the computation time and the accuracy of the converged solution. Nevertheless, the majority of recent UAV trajectory optimizations has focused on direct methods mainly because of their computational efficiency compared to the indirect methods. The computational efficiency of direct methods can be increased when they are combined with receding horizon strategy, particularly for the online applications.

Receding horizon trajectory optimization is another way to reduce the computational complexity wherein the optimization is concerned with only the local segment or a finite horizon of the entire path, and it computes the open-loop optimal control or path for that segment. The first part of the computed optimal control is taken for the control input to the system until a new optimal result arrives, and the procedure is repeated till the destination is reached. In fact, this type of optimization cannot guarantee the optimality of the resultant path globally, but it actually provides a nice way of achieving desired performance along with handling the nonlinear system with constraints; it is robust to uncertainties, which cannot be obtained by infinite horizon optimal control; nonetheless, the optimality of the resultant control and trajectory is suboptimal. This optimization scheme is the well-known model predictive control (MPC) or receding horizon control (RHC). MPC has received wide attention in the broad field of control theory for the stabilization of nonlinear systems with constraints. Mayne [87] made an insightful survey of different formulations of MPC and its notable stability issues. Jadbabaie [60] proved that augmentation of cost function with a special terminal cost function named Control Lyapunov Function (CLF) can guarantee the stability of RHC. Thus, with proven stability and appropriateness to nonlinear systems with constraints, MPC has been widely used in numerous trajectory planning problems [6, 27, 74, 78, 90, 118], especially for the obstacle avoidance problems in which the feasibility of the trajectory and the control of the vehicle have more priority than the optimality of the global path.

In general, direct methods have been used to solve the optimization of MPC whereas Schouwenaars [114] and other researchers have used mixed-integer linear programming (MILP) to solve an MPC problem for path planning of UAVs by discretizing the nonlinear problem into discrete mixed integer and linear programming. MILP is a powerful mathematical technique to solve linear programming problems having either integer variables or discrete logic. Because general path planning problems may include online discrete decision processes or discrete variables, MILP has been considered the primary optimization tool for dealing with such problems, and CPLEX [2] is the representative solution software for MILP. However, despite many successful stories, MILP may not be the proper choice for handling the computational complexity of trajectory planning in dynamically changing environments [114] because it requires another systematic or analytical formalization process that converts the obstacle environment into MILP formulation, which might be impossible for online implementation.

1.4 Real-time Nonlinear Optimization Techniques

Although the hardware performance of today's computers has been enormously enhanced recently, the computational complexity of the dynamic optimization problems has prevented the indirect optimization methods from being used for online applications, except in the special case of linear unconstrained systems with quadratic cost, that is, LQR. Most direct optimization methods have been invented to circumvent the difficulties of the indirect optimization for online applications. As described previously, direct methods discretize the system states and inputs by using collocation or approximation to convert the OCP to NLP and solve the NLP with well-established numerical solvers. Relevant selection of a discretization and an approximation can efficiently reduce the dimension of the converted NLP, resulting in increased computational efficiency in numerical optimization, thus significantly reducing computation time while not affecting the admissible accuracy of the solution.

One special way to reduce the dimension of the optimization problem is to exploit the concept of differential flatness of dynamic systems. If the dynamic system is *differentially flat*, the states and inputs can be directly expressed in terms of the specific outputs of the system, named the flat outputs and their derivatives [37]. All linear systems and the feedback linearizable nonlinear systems are typical differentially flat systems. It is well recognized that utilizing the flatness can increase computational efficiency. Numerical integration of the system equation is unnecessary to get the states and the inputs. One may effectively find a smooth curve of flat outputs that satisfies the system dynamics algebraically; therefore, the dimension of the problem can be significantly reduced from that of states and inputs to the smaller dimension of flat outputs. A well-known direct numerical optimization technique utilizing this differential flatness is the Nonlinear Trajectory Generation (NTG), developed by Milam [91] at Caltech. NTG uses the B-spline parameterization of flat output and collocation to convert the optimization problem to the NLP, thereby composing the NLP with variables of B-spline coefficients or control points, and it uses NPSOL [45], a well-known sequential quadratic programming (SQP)-based NLP solver. NTG can compute the optimal trajectory fast, and it has been used as the trajectory optimizer in other applications [58, 92, 95], the previous research [68, 105] of this thesis, and this thesis as well. However, it is only applicable to differentially flat systems, and unfortunately many nonlinear optimization problems, if they include complex system dynamics, cannot be merely treated as flat systems without any level of simplification of dynamics and constraints.

Unlike the method using differential flatness, the pseudo-spectral (PS) method is widely applicable to linear, nonlinear, flat, or even differential inclusion systems [35, 111], and it is gaining a great deal of attention recently. The PS method was originally developed for solving partial differential equations in fluid dynamics [17], and over the last few decades it has emerged as one of the major online optimization tools. PS methods directly convert OCP to NLP by pseudo-spectral approximations of states and inputs, then they use a numerical NLP solver. It is noted that PS approximation can provide relatively accurate approximations for smooth functions, integration, and differentiation, even with a small number of spectral nodes [8], and it guarantees an exponential convergence rate [4]. Those are crucial ingredients for solving a dynamic optimization problem in real time.

1.5 Thesis Contributions

The purpose of this thesis is to present an online trajectory planning framework based on trajectory optimization to guide rotary-wing UAVs for low-altitude flight in uncertain environments such as NOE flight. The study stemmed from past research outcomes on the time-optimal obstacle avoidance approaches in the vertical plane [68, 93, 94, 105]. This thesis describes the improvements of the previous works to *the real-time trajectory planning for UAVs in three-dimensional space by using a receding horizon (RH) trajectory optimization to generate the local optimal trajectory, minimizing deviations from the preplanned path as well as avoiding measured obstacles.* To accomplish this objective, some fundamental changes were made to the previous works: the overall framework was redesigned, the optimization formulation was changed, A LIDAR was interfaced to detect obstacles, an obstacle grid map generation algorithm was added, a climb rate limit detection logic was added to detect a saturation of climb rate during vertical maneuver, and a global path search algorithm using dynamic programming was added to provide an initial guess or a template path to the RH trajectory optimizer. The details of the contributions are listed as follows:

- The study of this thesis establishes an RH trajectory optimization scheme for obstacle avoidance in an uncertain obstacle field. This study adopts the similar architecture of trajectory generation and vehicle controller in [90, 98], i.e., the trajectory optimization works as the command trajectory generator by solving open-loop optimal trajectory within the finite horizon of the maximum sensor range, then the commands are followed by the vehicle controller. This scheme is used in many similar applications of MPC or RHC but the work here considers the following features: the use of a simplified model for the vehicle plus the flight controller in trajectory optimization, the optimization of three-dimensional avoidance trajectory over the measured obstacle geometry so that the problem includes nonlinear constraints, and the practical implementation of RH trajectory optimization by multi-threaded programming.

- This thesis focuses on the practical implementation of the integrated framework that can provide the three-dimensional obstacle field navigation capability for a rotary-wing UAV as well as vertical terrain following. For this purpose, a sensor, LIDAR, is used to measure the obstacles or terrain geometry in the vertical plane and the point cloud data from the sensor is sampled and mapped to a grid of finite area in front of the vehicle to construct the obstacle map while the vehicle is yawing. In the formulation of optimization problem, the safety clearance to the complex and unexpectedly changing obstacle geometry is formulated as a path constraint. This thesis uses basic filling and spatial filtering in the procedure of obstacle grid mapping, and the mapped grid is processed with a blob detection algorithm to build a obstacle cuboid, which is recorded in a database and used for the global path searching

- The path planning of this thesis is actually a hybrid method, an online receding horizon trajectory optimization in conjunction with a coarse global path searching, which is similar to the approaches in other recent literature on trajectory planning of UAVs with the combination of global path searching and heuristic local path planning[57, 107] or searching and MPC [78, 90]. However, the detailed approach of this thesis is slightly different from those previous works; the approach of this thesis considers an unknown environment, so both coarse global path searching and local trajectory planning are computed in real time without any offline processing or computation. For global path searching, this thesis uses the DP to search the global path to the destination on the cuboid obstacle field, which can be constructed online by default, provided before flight, or received from other vehicles which already traversed the area earlier. The coarse path is updated at every second, and the local trajectory optimization uses the coarse path as an initial guess to the optimization process or as the local template path to be followed. Use of this hybrid approach can improve the typical weakness of receding horizon trajectory optimization, the possibility of non-optimality in a global sense, and it can increase the computational efficiency of the local trajectory optimization by providing feasible initial guesses.

- The integrated framework proposed in this thesis was evaluated in simulations and flight tests using a rotary-wing UAV test-bed at Georgia Tech. For the flight test evaluation, the benchmark tests proposed by Mettler et al. [89] were conducted and the results were compared to the baseline optimal solutions in [89]. In addition, the 3D obstacle avoidance capability of the developed framework was demonstrated in a real world environment. The results from the flight test demonstration are presented and discussed in this thesis.

1.6 Thesis Outline

The previous subsections presented overviews of UAV systems, UAV autonomy, the future perspectives requiring autonomous maneuvers in uncertain or dynamic obstacle fields, legacy methods and new trajectory planning methods gaining attention for UAV applications, and the approach of this thesis to obstacle avoidance. In Chapter 2, preliminary mathematical background in conjunction with optimization will be presented. Chapter 3 will introduce the past studies on the time-optimal approaches, and the continuous vertical obstacle avoidance that can be used for an optimal terrain following. Chapter 4 will address a logic for climb rate limit detection which is studied to increase the safety of vertical obstacle avoidance. In Chapter 5, the final implementation of the framework for three-dimensional trajectory planning will be presented in detail and Chapter 6 will provide typical results obtained from simulations and flight tests. Finally, Chapter 7 will draw conclusions of this study and suggest some ideas for future work.

CHAPTER II

MATHEMATICAL BACKGROUND

2.1 Trajectory Optimization

The trajectory planning problem might correspond to the casual question, "*How we can make the vehicle safely move from A to B at the same time obtaining ...?*" In fact, the use of optimization techniques is the most natural way of answering such a question, for it provides a nice mathematical way of computing a trajectory for a dynamical system that fulfills the demanded objective while naturally taking into account the system dynamics and subjected constraints. For this reason, the trajectory planning problem has been a major subject for optimization. In this section, we will overview the mathematical background of the optimization methods for trajectory planning.

2.1.1 Generic Nonlinear Optimization Problem Formulation

Let the dynamical system under consideration be mathematically described with the nonlinear differential equation with the states $x \in \mathbb{R}^n$ and the inputs $u \in \mathbb{R}^m$.

$$\dot{x} = f(x, u) \tag{1}$$

where all the vectors and functions are real-analytic. We want to find the optimal trajectory, $x^*(t)$ and $u^*(t)$, of Equation (1) in $[t_0, t_f]$ that minimizes the cost functional

$$J = \phi_f(x_f, u_f, t_f) + \int_{t_0}^{t_f} L(x(t), u(t))dt \tag{2}$$

and the dynamical system is subjected to n_0 initial, n_f final, and n_t trajectory constraints respectively given as,

$$\begin{aligned} b_{l,0} \leq \quad & \psi_0(x_0, u_0) & \leq b_{u,0} \\ b_{l,f} \leq \quad & \psi_f(x_f, u_f) & \leq b_{u,f} \\ b_{l,t} \leq \quad & \psi_t(x(t), u(t), t) & \leq b_{u,t} \end{aligned} \tag{3}$$

where, the functions $\psi_0 : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n_0}$, $\psi_f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n_f}$, $\psi_t : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}_+ \to \mathbb{R}^{n_t}$ are assumed to be continuous and at least twice differentiable, C^2, and the final time t_f is either free or fixed. Practically, the dimension of the state of a three-dimensional trajectory optimization problem will be 9, with the position denoted $p(t) \in \mathbb{R}^3$, the velocity $v(t) \in \mathbb{R}^3$, and acceleration vector $a(t) \in \mathbb{R}^3$ in the 3D inertial coordinate frame, and the acceleration is usually the input.

2.1.2 Optimal Control Problem

Perhaps the most widely-exploited method for addressing trajectory optimization is formulating the optimization problem as an optimal control problem [14, 15], which finds the optimal solution from the necessary conditions using the calculus of variations by forming a controlled Hamiltonian

$$H(x, u, \lambda) = L(x, u) + \lambda^T f(x, u) \tag{4}$$

where $\lambda \in \mathbb{R}^n$ is the co-state vector. Under the assumption that the Hamiltonian is continuously differentiable, Pontryagin [103] showed that the optimal control minimizes the Hamiltonian with respect to all inputs of admissible set, $u \in \mathcal{U}$.

$$u^* = \underset{u \in \mathcal{U}}{\operatorname{argmin}}\, H(x^*, \lambda^*, u) \tag{5}$$

and along with the initial condition, the final condition, and the necessary condition of the optimality

$$\dot{x} = H_\lambda \tag{6}$$

$$\dot{\lambda} = -H_x \tag{7}$$

$$0 = H_u \tag{8}$$

$$\lambda^T(t_f) = \phi_{f_x}|_{t=t_f} \tag{9}$$

form the two point boundary value problem and if the final time, t_f, is free, we should include the condition

$$(\dot{\phi} + H)|_{t_f} = 0 \tag{10}$$

Therefore, the optimal trajectory can be completely determined by solving the above differential Equations (5) through (10) by the numerical methods introduced in [9]. This is the approach taken by most indirect methods. However, although the above equations provide a computable solution to the problem, the optimal control approach is not practical for the real-time control of non-trivial systems as stated in the introduction. For this reason, other optimization methods have been proposed to solve the problem of optimization for the nonlinear dynamical systems, for example, the direct method.

2.1.3 Nonlinear Programming

The previous section states that the trajectory generation problem can be formulated as the optimal control problem (OCP). In general, the OCP can be solved by either indirect or direct methods. Indirect methods are based on the calculus of variations and the maximum principle solving Equations (5) through (10) numerically; otherwise the direct method solves the optimization problem by transforming the OCP into a nonlinear programming problem (NLP).

The generic NLP can be stated as

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ s.t.\quad b_{l_i} \leq &g_i(x) \leq b_{u_i}, \quad i = 1, \cdots, l \end{aligned} \tag{11}$$

where $f(x) : \mathbb{R}^n \to \mathbb{R}$ is the nonlinear cost function to minimize, $g_i(x) : \mathbb{R}^n \to \mathbb{R}$, $b_{l_i}, b_{u_i} \in \mathbb{R}$ are the nonlinear constraint function, lower, and upper boundaries respectively. The cost and constraint functions are at least C^2.

Various numerical methods for finding the local optimum of the general NLP problem have been well developed and implemented in the software solutions. NPSOL and SNOPT are the sequential quadratic programming (SQP)-based solvers developed by Gill et al. [45, 46] and they are perhaps the most widely used solvers for general NLP problems in many applications. CONOPT [34] is a recent solver based on the generalized reduced gradient algorithm, MINOS [97] uses the projected Lagrangian method to linearize the nonlinear constraints and search for an optimal solution in an augmented Lagrangian form, RIOT [115] uses the adjoint method, and IPOPT [70] is the large scale sparse NLP solver based on the interior point method (IPM). IPMs are another popular class of methods known to be effective and reliable for solving local NLP solutions.

2.1.4 Sequential Quadratic Programming

SQP methods have proved highly effective for solving constrained optimization problems with continuous nonlinear functions of the objective and constraints and have been particularly successful in solving the optimization problems arising in optimal trajectory calculations [46]. The idea of SQP is to solve iteratively the NLP using a sequence of quadratic programming (QP) sub-problems at any given sequence of x_k, which is *major iteration*, and then uses the solution of the sub-problem to construct a new iterate x_{k+1}, which is *minor iteration.* The iterative procedure is done in such a way that the sequence x_k converges to a local minimum x^* of the NLP of Equation

(11) as $k \to \infty$.

The NLP formulation of Equation (11) generally can be expressed with the standard form

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ &s.t.\ c(x) \geq 0 \end{aligned} \tag{12}$$

where $c(x) \in \mathbb{R}^m$ and m is the number of constraints in the standard form. Now, we introduce the modified Lagrangian of the NLP as

$$\mathcal{L}(x, \lambda) = f(x) - \lambda^T c(x) \tag{13}$$

and then suppose there exists an optimal solution of the NLP, $x^* \in \mathbb{R}^n$, and the Lagrange multiplier λ^*, satisfying the first-order Karush-Kuhn-Tucker (KKT) necessary optimality conditions:

$$\begin{aligned} c(x^*) &\geq 0 \\ \lambda^* &\geq 0 \\ c(x^*)^T \lambda^* &= 0 \\ J(x^*)^T \lambda^* - g(x^*) &= 0 \end{aligned} \tag{14}$$

where $J(\cdot)$ is the Jacobian of $c(\cdot)$ and $g(\cdot)$ is the gradient of $f(\cdot)$ at any vector x. Note that x^* is a stationary point of the optimization problem and it is not necessarily an unconstrained minimizer of the Lagrangian, unless it satisfies the second order conditions:

1. The columns of J are linearly independent

2. Strict complementary slackness holds at x^*

3. The Hessian of the Lagrangian with respect to x is positive definite on the null space of $J(x^*)$, i.e.,

$$\begin{aligned} &x^T \nabla^2 \mathcal{L}(x^*, \lambda^*) x > 0 \\ &x \in \mathbb{R}^n,\ x \neq 0,\ J(x^*)^T x = 0 \end{aligned} \tag{15}$$

Both the first order KKT necessary conditions and the second order necessary conditions comprise the second order sufficient optimality conditions of the NLP.

The construction of the QP sub-problem is the same as the local quadratic approximation of the cost and the constraints at x_k

$$\begin{aligned} f(x) &\approx f(x_k) + g(x_k)(x - x_k) + \tfrac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k) \\ c(x) &\approx c(x_k) + J(x_k)(x - x_k) \end{aligned} \tag{16}$$

and this approximation leads to the following QP subproblem

$$\begin{aligned} &\min g(x_k)^T p + \tfrac{1}{2} p^T \nabla^2 f(x_k) p \\ &s.t. \;\; c(x_k) + J(x_k)^T p + d = 0 \end{aligned} \tag{17}$$

where $p = x - x_k$ and $d \in \mathbb{R}^m$ is the slack variable

In fact, the second order sufficient condition for optimality implies that x^* is the local minimizer of the problem:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda^*) \\ &s.t. \; c(x) \geq 0 \end{aligned} \tag{18}$$

and by the quadratic approximation of $\mathcal{L}$

$$\mathcal{L}(x, x_k, \lambda_k) \approx \mathcal{L}(x_k, \lambda_k) + \nabla \mathcal{L}(x_k, \lambda_k)^T p + \frac{1}{2} p^T \nabla^2 \mathcal{L}(x_k, \lambda_k) p \tag{19}$$

The equivalent NLP of Equation (18) can be approximated to the QP sub-problem:

$$\begin{aligned} &\min \nabla \mathcal{L}(x_k, \lambda_k)^T p + \tfrac{1}{2} p^T \nabla^2 \mathcal{L}(x_k, \lambda_k) p \\ &s.t. \;\; c(x_k) + J(x_k)^T p + d = 0 \end{aligned} \tag{20}$$

and both QP sub-problems of Equations (17) and (20) are equivalent from the necessary conditions of optimality.

In fact, the exact calculation of the Hessian of Lagrangian $\nabla^2 \mathcal{L}$ might not be trivial; usually it is difficult to get the Hessian, especially in a large dimensional problem and it significantly affects the numerical convergence and speed of the overall SQP method. Most SQP methods use the quasi-Newton approximation of the Hessian

$$H_k \approx \nabla^2 \mathcal{L}(x_k, \lambda_k) \tag{21}$$

and a key challenge to developing a fast algorithm for SQP is to find an accurate approximation method. The most popular quasi-Newton method many SQP solvers are adopting is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update method.

$$H_{k+1} = H_k + \frac{yy^T}{y^T s} - \frac{B_k s s^T B_k}{s^T B_k s} \tag{22}$$

where $s = x_{k+1} - x_k$, $y = \nabla\mathcal{L}(x_{k+1}, \lambda_{k+1}) - \nabla\mathcal{L}(x_k, \lambda_k)$, therefore the final form of approximate QP sub-problem is given as

$$\begin{aligned} &\min g(x_k)^T p + \tfrac{1}{2} p^T H_k p \\ &s.t.\ \ c(x_k) + J(x_k)^T p + d = 0 \end{aligned} \tag{23}$$

Finding the solution of the above QP sub-problem of Equation (23) is another iterative procedure as stated before. Both NPSOL and SNOPT use SQOPT which is based on solving a sequence of linear systems involving the reduced Hessian $Z^T H_k Z$, where Z is defined implicitly using the sparse LU factorization. Reduced-Hessian methods are known as best suited to problems with small degrees of freedom, i.e., problems for which many constraints are active [46].

After a QP sub-problem has been solved, SQP methods perform minor iterations to find new estimates of the QP solution. The classical method is the Newton method which has quadratic convergence rates but is usually sensitive to the initial point and fails to converge. More general gradient descent methods converge from nearly any starting point but have poor local convergence rates due to the problem of low gradient. Most practical NLP solvers are adopting line search methods that ensure robust convergence. Line search methods usually vary the step size along the search direction decided from the current point, as given by this general notation

$$x_{k+1} = x_k + \alpha_k \cdot p_k \tag{24}$$

where α_k is the step length and p_k is the search direction vector which can be decided with various approaches at every major iterate x_k. For general unconstrained minimization, the best step length α_k^* is the minimizer of the objective cost function $f(x_{k+1})$ in the search direction and in some trivial cases, the line search method can get one shot convergence to the minimum but usually determining a minimizer along p_k is iterative and frequently time consuming.

Another main branch of methods used in minor iteration is the trust region method. The basic idea of the trust region method is that it changes the region of the search, that is, the trust region, depending on how well the local quadratic model

$$q_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T H_k p \tag{25}$$

matches the actual function evaluation. Typically, the trust region is taken to be an ellipse that can be decided by the eigen-state of the approximate Hessian H_k. Once the step based on the quadratic model lies inside the trust region, then the region is trusted to have the minimum, the step is chosen, and the trusted region is decreased. On the other hand, if the step based on the model lies outside the region, the step is taken just to the boundary of the region, then the region is enlarged. Once the step is chosen, the ratio of the difference of actual function evaluation and the difference of the model value is computed to decide the direction of trust region modification and whether to enlarge or shrink, starting from the initial radius given as Δ.

Once the solution of the QP sub-problem (x_k, λ_k) is obtained, the estimates of the solution (x_{k+1}, λ_{k+1}) are determined by a line search or trust region method from the current solution. However, regardless of whether a line search or trust region method is used, when the feasibility of the iterates is not maintained, it can be difficult to choose the proper step length and may result in the failure to converge. Especially when the problem includes any nonlinear constraint, maintaining feasibility at every iteration becomes difficult [91]. When such infeasible conditions happen, the chosen step length should minimize the objective function at the same time reducing the infeasibilities of the constraints. Merit functions used in NPSOL and SNOPT, which is the augmented Lagrange function, are used to guide the improvement of the feasibility and the optimum at the same time.

$$\mathcal{M}_q(x, \lambda) = f(x) - \lambda^T c(x) + \frac{\rho}{c}(x)^T c(x) \tag{26}$$

where $\rho > 0$ is the penalty value for weighting constraints variation and λ is the estimated set of the Lagrange multipliers. The proper step length should sufficiently decrease the merit function.

2.1.5 Conversion of OCP to NLP

As stated before, the optimal trajectory planning problem which is formulated with the form of OCP can be converted to the NLP problem, and after conversion, the NLP can be solved efficiently with the solution technique overviewed in the previous section, the SQP method. There are many conversion techniques for various direct optimization methods and some of the widely used conversion techniques are well surveyed by Betts [9] and Conway [24].

2.1.5.1 Direct Collocation

One reliable technique to convert OCP to NLP is the collocation method. Dickmanns [30] used the collocation scheme to solve the TPBVP of the indirect method and it is significantly more robust than the shooting method in solving TPBVP. However, the most useful ideas of the collocation method, known as *direct collocation* were outlined by Hargraves and Paris [52] and Stryk [124].

The first step in the collocation method is to break the time domain into small piecewise intervals of times.

$$t_0 = t_1 < \cdots < t_N = t_f \tag{27}$$

The parameter y of nonlinear programming is the vector of control and states at the discrete time points, $t_i, i = 1, \cdots, N$ and the final time t_f:

$$y = [u(t_1), \cdots, u(t_N), x(t_1), \cdots, x(t_N), t_N]^T \in \mathbb{R}^{N(m+n)+1} \tag{28}$$

In general collocations, the input is approximated to be piecewise linear

$$u_i(s) = u(t_i) + s(u(t_{i+1}) - u(t_i)) \tag{29}$$

where $s = t/(t_{i+1} - t_i) \in [0, 1]$ and the states are approximated with the Hermite cubic polynomial between discrete points defined in terms of the endpoint values and first derivatives at the endpoints.

$$\begin{aligned} x_i(s) = &[2(x(t_i) - x(t_{i+1})) + \dot{x}(t_i) + \dot{x}(t_{i+1})]s^3 \\ &+ [3(x(t_{i+1}) - x(t_i)) - 2\dot{x}(t_i) - \dot{x}(t_{i+1})]s^2 \\ &+ \dot{x}(t_i)s + x(t_i)) \end{aligned} \tag{30}$$

To ensure the approximation accurately represents the system differential equation, the derivative of the midpoint of each polynomial segment, $\dot{x}_{c_i} = \dot{x}_i(0.5)$, is compared to the evaluation of the system equation using the interpolated states and the inputs, $f(x_{c_i}, u_{c_i})$. The slopes at the collocation points are

$$\dot{x}_{c_i} = -\frac{3}{2(t_{i+1} - t_i)}(x(t_i) - x(t_{i+1})) - \frac{1}{4}(\dot{x}(t_i) + \dot{x}(t_{i+1})) \tag{31}$$

and the *defect* is defined as given, thus turning the system dynamic Equation (1) into additional constraints to be bounded at every node

$$\xi = f(x_{c_i}, u_{c_i}) - \dot{x}_{c_i} \tag{32}$$

Hargraves and Paris [52] considered the Mayer type cost functions only. If the problem is a Bolza problem, the problem with terminal plus integral cost, direct methods need a numerical or analytical quadrature method to integrate the integral cost. If the cost function is not complex, analytical integration may be simplified by taking advantage of the piecewise cubic polynomials. However, when the cost function is complicated or highly nonlinear, an analytic expression for the integral may be too difficult to derive. In this situation, numeric quadrature such as trapezoidal, Simpson, or Gauss are more appropriate. Accuracy of the integration is adjusted by dividing up each segment into a given number of sub-nodes but this directly affects the time required for the solution.

The above transcription method is the most common one but there are many variations of the direct collocation differing primarily on how the implicit integration rules are constructed. In fact, recent pseudo-spectral approaches are also kinds of direct collocations.

2.1.5.2 Pseudo-spectral Method

The Pseudo-spectral (PS) method is another branch of direct collocation techniques that uses pseudo-spectral collocations and the implicit approximation of the integration and the differentiation using orthogonal polynomials, also known as PS approximation. It is widely recognized that PS methods have advantages with regard to accuracy because they use more accurate approximating polynomials compared to general direct collocation methods, and it is possible to use fewer (and thus wider) segments, which yields a much smaller NLP problem and obtains the same accuracy in the solution. The word *spectral* refers to the error convergence rate with respect to an increasing number of nodes, and spectral convergence means that error decreases faster than the rate of $\mathcal{O}(N^m)$ for any $m > 0$, simply meaning that error decreases exponentially with increasing N. The variants of pseudo-spectral methods use different discretization schemes for the collocation points and the interpolating functions, and are generally named for the scheme used. The Legendre pseudo-spectral method uses the Legendre interpolating polynomial and the Legandre-Gauss-Lobatto (LGL) collocation nodes, and the Chebyshev pseudo-spectral method uses the Chebyshev polynomial and Chebyshev-Gauss-Lobatto (CGL) collocation nodes; otherwise the Gauss pseudo-spectral method uses the Legendre polynomial and the Legendre-Gauss (LG) collocation. The Gauss pseudo-spectral method differs from several other pseudo-spectral methods in that the dynamics are not collocated at the boundary points, which leads to the proper and accurate approximation to the costate [8]. Details about the spectral collocation methods and the variation of PS methods are well explained by Garg et al. [42].

Currently, PS methods are well established for the various OCP applications in different forms of implementations and software. Software packages such as DIDO [110] and GPOPS [106] are representative PS optimization tools which run in MATLAB®. PSOPT [4] is an open source software library written in C++.

In the Legandre PS method, time is directly discretized with LGL nodes which includes both endpoints of the time, that is, $t \in [t_0, t_f] \rightarrow \tau \in [-1, 1]$, using the mapping relation given by

$$t = \frac{(t_f - t_0)}{2}\tau + \frac{(t_f + t_0)}{2} \tag{33}$$

and by this mapping, the system of Equation (1) and the cost of Equation (2) can be transformed as given

$$\frac{2}{(t_f - t_0)}\frac{dx}{d\tau} = f(x(\tau), u(\tau)) \tag{34}$$

$$J = \phi_f(x(1), u(1)) + \int_{-1}^{1} L(x(\tau), u(\tau))d\tau \tag{35}$$

Legandre PS methods and Gauss PS methods uses Legendre polynomials of degree N given by

$$L_i(\tau) = \prod_{\substack{j=0 \\ j\neq i}}^{N} \frac{\tau - \tau_i}{\tau_i - \tau_j}, \quad 0 \le i \le N \tag{36}$$

otherwise Chebyshev PS methods use Chebyshev polynomial of degree N at CGL nodes $\tau_i = -\cos(\pi i/N)$ in $[-1, 1]$

$$C_i(\tau) = \cos(i \cos^{-1}(\tau)), \quad 0 \le i \le N \tag{37}$$

In the Legendre PS method, the states and inputs are approximated with the polynomial

$$x(\tau) \approx x^N(\tau) = \sum_{i=0}^{N} x_i L_i(\tau) \tag{38}$$

$$u(\tau) \approx u^N(\tau) = \sum_{i=0}^{N} u_i L_i(\tau) \tag{39}$$

where x_i and u_i are given values at node point i. The derivatives of the states can be approximated by differentiating Equation (38)

$$\dot{x}(\tau) \approx \dot{x}^N(\tau) = \sum_{i=0}^{N} x_i \dot{L}_i(\tau) = \sum_{i=0}^{N} D_{ki} x_i \tag{40}$$

where D_{ki} is the differentiation matrix given by

$$D_{ki} = \dot{L}_i(\tau_k) = \sum_{i=0}^{N} \frac{\prod_{\substack{j=0 \\ j\neq i}}^{N} \tau_k - \tau_i}{\prod_{\substack{j=0 \\ j\neq i}}^{N} \tau_i - \tau_j} \tag{41}$$

and evaluating the expression at the LGL nodes gives the $(N+1) \times (N+1)$ differentiation matrix $\mathbf{D}$

$$\mathbf{D} = \begin{cases} \frac{L_N(\tau_k)}{L_N(\tau_i)} \frac{1}{\tau_k - \tau_i}, & k \neq i \\ -\frac{N(N+1)}{4}, & k = i = 0 \\ \frac{N(N+1)}{4}, & k = 1 = N \\ 0, & \text{otherwise} \end{cases} \tag{42}$$

thus, the system Equation (34) and the cost Equation(35) can be approximated

$$\dot{x}(\tau_k) \approx \sum_{i=0}^{N} D_{ki} x_i = \frac{t_f - t_0}{2} f(x_k, u_k) \tag{43}$$

$$J \approx \phi(x_N, u_N) + \frac{t_f - t_0}{2} \sum_{i=0}^{N} w_i L(x_i, u_i) \tag{44}$$

where the Gauss weights, w_i, is defined as

$$w_i = \frac{2}{N(N+1)} \frac{1}{(L_N(\tau_k))^2} \tag{45}$$

Therefore the NLP problem can be formed by Equations (43) and (44) approximated from the OCP of Equations (1) through (3)

$$\begin{aligned}
\min_{x_i,u_i} J = \phi(x_N, u_N) + \frac{t_f - t_0}{2}\sum_{i=0}^{N} w_i L(x_i, u_i) \\
\text{subject to} \quad \sum_{i=0}^{N} D_{ki}x_i - \frac{t_f - t_0}{2} f(x_k, u_k) = 0 \\
b_{l,0} \le \psi_0(x(-1), u(-1)) \le b_{u,0} \\
b_{l,f} \le \psi_f(x(1), u(1)) \le b_{u,f} \\
b_{l,\tau_i} \le \psi_t(x(\tau_i), u(\tau_i), \tau_i) \le b_{u,\tau_i}
\end{aligned} \tag{46}$$

having the unknown variables composed of $x_i, u_i,\ 0 \le i \le N$.

2.1.5.3 Adjoint Method

The other way to convert OCP to NLP is the adjoint method. The adjoint method is a direct method that uses a combination of nonlinear programming and shooting of adjoint variables. In contrast to the collocation method, the adjoint method has a significantly lower number of decision variables $u(t_i)$. In fact, collocation methods usually require many collocation points to generate usable controls. The adjoint method uses the backward integration of the adjoint (or costate) system to determine the gradients of the cost and constraints by the control that significantly contributes to the robust solution of Equation (11). Bryson and Ho [15] state that numerical integration of the adjoint variable is quite stable since integration is carried out in backward time, assuming that the system is stable in forward time. Applying the adjoint lemma and constructing control deviation history $\delta u(t)$ such that the cost function is decreasing, the gradient of the cost function of Equation (2) can be determined by taking the derivative of the cost

$$\begin{aligned}
\nabla_u J(u) = & \lambda_c^T f_u + L_u \\
\dot{\lambda} = -f_x^T \lambda - L_x^T, &\ \ \lambda(t_f) = \frac{\partial \phi(x(t_f))}{\partial x(t_f)}
\end{aligned} \tag{47}$$

the gradient of the end point constraint, and the gradient of the state inequality constraint can be obtained by a similar manner and detailed expressions are shown in [91].

2.1.6 Differential Flatness of Dynamic Systems

Differential flatness of dynamic systems first introduced and studied by Fliess et al. [37] provides the efficient way of controlling nonlinear dynamical systems in real time. If the dynamical system is *differentially flat*, it is possible to find a set of flat outputs

$$z = \mathcal{C}(x, u, \dot{u}, \cdots, u^{(\gamma)}), z \in \mathbb{R}^m \tag{48}$$

with which the states and the input can be directly expressed algebraically:

$$x = \mathcal{A}(z, z, \dot{z}, \cdots, z^{(\alpha)}) \tag{49}$$

$$u = \mathcal{B}(z, z, \dot{z}, \cdots, z^{(\beta)}) \tag{50}$$

All linear systems and the feedback linearizable nonlinear systems are typical differentially flat systems. It is well recognized that utilizing the flatness can increase computational efficiency. First, numerical integration of the system equation to get the states and the inputs is unnecessary. Secondly, one may effectively find a smooth curve of flat outputs that satisfies the system dynamics algebraically. Therefore the dimension of the problem can be significantly reduced from that of states and inputs to the smaller dimension of flat outputs.

Mathematically, the change of the states and the input, Equations (49) and (50), will linearize the system Equation (1) into the trivial system written in Brunovsky canonical form [37, 67]

$$\begin{cases} z_1^{(\nu_1)} &= \nu_1 \\ & \vdots \\ z_m^{(\nu_m)} &= \nu_m \end{cases} \tag{51}$$

where $\nu_1 = \dot{u}_1, \cdots, \nu_m = \dot{u}_m$ are controllability indices and $z_1, \cdots, z_m$ are flat outputs, thus, the system behavior can be expressed without the integration of the system

Equation (1) by the flat outputs and its finite number of derivatives. This means that finding the trajectory from $x(0), u(0)$ to $x(T), u(T)$ is changed to finding any smooth curve that connects $z^k(0)$ and $z^k(T)$ up to some finite numbers; then the control input can be recovered by the Equation (50) . Therefore, it is unnecessary to solve the TPBVP for optimal solution if the system is flat.

This general idea can be traced back to works by D. Hilbert and E. Cartan on under-determined systems of differential equations, where the number of equations is strictly less than the number of unknowns. It is an arguable fact that this property may be extremely useful when dealing with trajectories: from z trajectories, x and u trajectories are immediately deduced [37].

The point mass approximation of vehicle dynamics that the majority of trajectory optimization problems use is a typical differentially flat system, if we choose the positions and time as the flat output. Unfortunately though, many nonlinear control systems may not be easily determined to be flat systems and no general method is available yet to ascertain the differential flatness of a given system. It is debatable whether or not the necessary and sufficient conditions for differential flatness exist. Fliess et al. [37] introduced necessary conditions and Charlet et al. [18] provided sufficient conditions for a class of systems. Moreover, even for differentially flat systems, currently there is no straightforward way of taking into account the flight envelope constraints [38]

2.1.7 Parameterization of Flat Output

The previous section introduced that the flatness of a dynamic system can eliminate the dynamic constraints and thus improve computational efficiency by reducing the overall problem dimension in finding the control of the system or in the optimization problem. If a nonlinear system is flat, the control problem turns out to be a problem of selecting the flat outputs and finding the smooth curves that can efficiently approximate the flat outputs. There are many curves that can be used for the approximation; Fourier series, polynomials, rational segments, and etc, can be used.

The approximation method should be able to accurately represent the curve of the flat output with a reasonable number of decision variables and should be able to set the curve to have a level of continuity C^k without adding additional conditions, i.e., constraints. Local support is also a demanded property of the approximation that means the change of the control variable of the curve for the interest of local modification should influence only locally. An approximation method that meets these requirements is piecewise Bezier polynomials or B-splines. An overview of B-splines, from which much of the following is derived, can be found in Deboor [28]. A B-spline curve is constructed from piecewise Bezier curves joined together with a prescribed level of continuity between them. The points at which the piecewise curves are joined are called *breakpoints* or *knot points*. The non-decreasing sequence of real numbers assigned to the breakpoints is called the *knot vector*. The *smoothness* s_i of a breakpoint stands for the level of continuity at the breakpoint such that a break point is C^{s_i-1} continuously differentiable. The order of piecewise curve k_i, smoothness, and multiplicity m_i have the following relation and Figure 6 shows an example curve constructed by a given spline entity.

$$k_i = m_i + s_i \tag{52}$$

and the required number of control points or B-spline coefficients are determined as given

$$p_i = l_i(k_i - s_i) + s_i \tag{53}$$

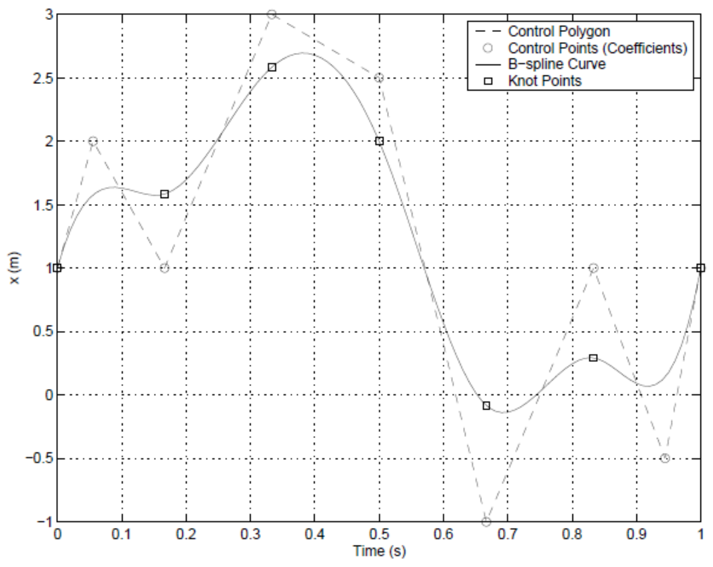


Figure 6: An example of B-spline curve: 6 intervals ($l = 6$), fourth order ($k = 4$), and C^3 continuity at breakpoints (i.e., smoothness s=3); Nine control points are required to meet these properties and to be the decision variables [91].

The outputs can be approximated by the B-spline relation

$$\begin{aligned} z_1(t) &= \sum_{i=1}^{p_1} B_{i,k_1}(t) C_i^1 \\ z_2(t) &= \sum_{i=1}^{p_2} B_{i,k_2}(t) C_i^2 \\ &\vdots \\ z_m(t) &= \sum_{i=1}^{p_m} B_{i,k_m}(t) C_i^m \end{aligned} \tag{54}$$

where p_j is the number of control points of the output z_j, C_i^j is the control points of the output z_j, and B_{i,k_j} is the basis functions of i-th control point for the j-th output given by the following recurrence relation.

$$\begin{aligned} B_{i,0}(t) &= \begin{cases} 1, \text{ if } t_i \le t \le t_{i+1} \\ 0, \text{ otherwise} \end{cases} \\ B_{i,k}(t) &= \frac{t - t_i}{t_{i+k+l} - t_i} B_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(t) \end{aligned} \tag{55}$$

The derivatives of the flat output can be derived from differentiating B-spline

Equation (54)

$$z_j^{(r)} = \sum_{i=0}^{p_j} B_{i,k_j}^{(r)}(t) C_i^j \tag{56}$$

where the r-th derivative of the basis function is given by

$$B_{i,k_j}^{(r)}(t) = \frac{k-1}{k-i-r}\left[\frac{t-t_i}{t_{i+k+1}-t_i} B_{i,k_j-1}^{(r)}(t) + \frac{t_{i+k}-t}{t_{i+k}-t_{i+1}} B_{i+1,k_j-1}^{(r)}(t)\right] \tag{57}$$

The B-spline parameters, i.e., l_i, k_i, s_i, need to be selected carefully to represent the trajectory of the output sufficiently accurately while maintaining computational efficiency. Using an increased value for the interval and order may accurately reconstruct the output history, but it also increases the computational load in the iterative procedure of optimization. Thus, relevant trade-off should is done depending on the allowable level of accuracy the problem requires.

2.2 Nonlinear Trajectory Generation (NTG)

Nonlinear Trajectory Generation (NTG), which was developed by Milam [91] at CalTech, is the direct optimization software library designed to solve the optimization problem of the constrained nonlinear systems, which is differentially flat, in real time. NTG provides the functionality to perform transcribing the cost and the constraints of the optimization problem with manually designated flat outputs of the system, it conducts the parameterization of the flat outputs with B-splines with given B-spline parameter sets, and finally performs the NLP solving for the B-spline coefficients that minimize the cost subject to constraints. NTG uses NPSOL for the NLP solver, so the main roles of NTG are converting the optimization problem into NLP with B-spline parametrization and providing functions and interfaces to develop the customized optimization software for the user optimal problem. The programming language NTG uses is C++ and the Fortran-based NPSOL is needed to be linked to the user software.

From the summary of the flatness of the system, if a nonlinear system is differentially flat, we may find the output

$$z = h(x), \ z \in \mathbb{R}^m \tag{58}$$

such that the states and the inputs can be recovered from the outputs and its derivatives

$$x = a(\xi), \quad u = b(\xi) \tag{59}$$

where the flat output vector ξ is

$$\xi = [z_1, \cdots, z_1^{r_1}, \cdots, z_m, \cdots, z_m^{r_m}]^T \tag{60}$$

then the optimization problem Equations (2) and (3) can be reformulated with the flat outputs

$$\begin{aligned} &\min_{\xi} J = \phi(a(\xi), b(\xi)) + \int_{t_0}^{t_f} L(a(\xi), b(\xi))dt \\ &\text{subject to} \quad b_{l,0} \leq \psi_0(a(\xi_0), b(\xi_0)) \leq b_{u,0} \\ &\qquad\qquad\quad b_{l,f} \leq \psi_f(a(\xi_f), b(\xi_f)) \leq b_{u,f} \\ &\qquad\qquad\quad b_{l,t_i} \leq \psi_t(a(\xi), b(\xi), t_i) \leq b_{u,t_i} \end{aligned} \tag{61}$$

and the flat output vector ξ can be determined from the B-spline parametrization by Equations (54) through (57) and the initial and final values, ξ_0, ξ_f, are found from the states and inputs x_0, x_f.

NTG in fact uses the collocation of time give as $d_c = t_0, \cdots, t_N$ and N is the number of collocation points and the quadrature to approximate the integration $I = \int_{t_0}^{t_p} \bar{L}(t)dt$.

$$I \approx \hat{I} = \sum_{i=0}^{q} \mu_i \bar{L}(t_i) \tag{62}$$

where the weight μ_i is determined in advance. The number of quadrature q is decided from the demanded convergence rate $\mathcal{O}(q^{-r})$ for some integers $r \geq 1$.

From Equations (54) through (57) and the collocation points, NTG builds the sparse collocation matrix that maps the coefficients of the B-splines to the flat output and the derivatives at every collocation point.

$$Z_i(t) = \begin{bmatrix} z_1^{(0)}(t_0)\\ \vdots \\ z_1^{(r_i)}(t_0)\\ z_i^{(0)}(t_1) \\ \vdots \\ z_i^{(r_i)}(t_1)\\ \vdots \\ z_i^{(0)}(t_N)\\ \vdots\\ z_i^{(r_i)}(t_N)\end{bmatrix} = \begin{bmatrix} X & & & & & & & & & & & \\ X & X & & & & & & & & & & \\ \vdots & & \ddots & & & & & & & & & \\ X & X & \cdots & X & & & & & & & & \\ \vdots & & \vdots & \vdots & & & & & & & & \\ X & X & \cdots & X & & & & & & & & \\ & & & & X & X & \cdots & X & & & & \\ & & & & \vdots & & \vdots & \vdots & & & & \\ & & & & X & X & \cdots & X & & & & \\ & & & & & & & & \ddots & & & \\ & & & & & & & & X & X & \cdots & X \\ & & & & & & & & \vdots & & \vdots & \vdots \\ & & & & & & & & X & X & \cdots & X \\ & & & & & & & & & & & X \\ & & & & & & & & & & X & X \\ & & & & & & & & & \ddots & & \vdots \\ & & & & & & & & X & X & \cdots & X \end{bmatrix} \begin{bmatrix} C_1^i \\ C_2^i \\ \vdots \\ C_{k_i-s_i}^i \\ \vdots \\ C^i_{2(k_i-s_i)} \\ \vdots \\ C^i_{p_i}\end{bmatrix} \tag{63}$$

By letting

$$\xi(t_i) = [Z_i(t_i), Z_2(t_i), \cdots, Z_m(t_i)]^T \tag{64}$$

finally, Equation (61) can be transformed to NLP

$$\begin{aligned} \min_U F(U) \approx \phi(a(\xi(t_f)), b(\xi(t_f))) + \sum_{\substack{j=0\\ j=j+q}}^{N} \sum_{kj=0}^{q} \mu_k L(a(\xi(t_{k+j})) \\ \text{subject to} \quad & b_{l,0} \leq \psi_0(a(\xi_0), b(\xi_0)) \leq b_{u,0} \\ & b_{l,f} \leq \psi_f(a(\xi_f), b(\xi_f)) \leq b_{u,f} \\ & b_{l,t_i} \leq \psi_t(a(\xi(t_i)), b(\xi(t_i)), t_i) \leq b_{u,t_i} \end{aligned} \tag{65}$$

where the control variable of the NLP is the B-spline coefficients vector U given by

$$U = \left[C_1^1 \cdots C_{p_1}^1, C_1^2 \cdots C_{p_2}^2, \cdots, C_1^m \cdots C_{p_m}^m\right]^T \tag{66}$$

NTG uses NPSOL to find the optimal solution U^* of the NLP problem given by Equation (65). The resultant optimal flat outputs, states, and the inputs can be recovered by the collocation given by Equations (63) and (64) and the flat output of Equation (59) such that

$$\begin{aligned} \xi^* &= G \cdot U^* \\ x^* &= a(\xi^*) \\ u^* &= b(\xi^*) \end{aligned} \tag{67}$$

2.3 Summary

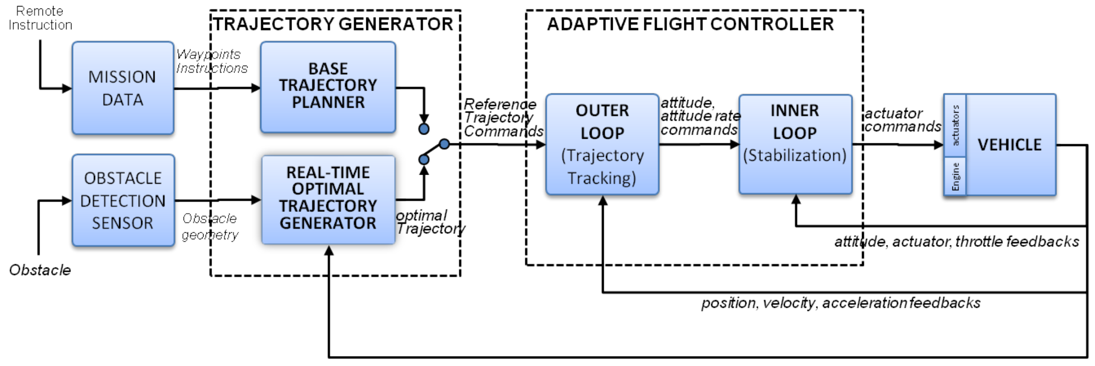
This chapter presented an overview of the mathematical foundation of the approach this thesis is based on, the optimal control problem (OCP) for nonlinear dynamical systems with constraints. In general, OCP does not have a closed-form solution unless the system is simple and unconstrained, and hence, numerical methods such as multiple shooting and relaxation techniques are often employed for the solution. The classical solution method for OCP is the use of Pontryagin's maximum principle and the induced necessary conditions, which is known as the indirect method. It is a well-known fact that accurate solutions can be obtained; however, the indirect methods are less robust to a poor initial guess, they present difficulties in dealing with the initial guess of co-states, and the computational load is too high for real-time applications. For this reason, direct methods have been mostly employed for real-time applications. Direct methods solve the optimization problem through nonlinear programming (NLP) having the objective function and the constraints converted from the original OCP. The sequential quadratic programming (SQP), which is the most popular numerical NLP solution technique and the one employed in this thesis, was reviewed and the methods of conversion were introduced: direct collocation, the adjoint method, and the recently developed pseudo-spectral method. The direct method this thesis employed is the Nonlinear Trajectory Generation (NTG), which is a spline-based direct method utilizing the differential flatness of the nonlinear system. The mathematical background of the differential flatness was introduced, and the spline representation of the flat output and the overall OCP transcription to NLP by the collocation was presented to help better understand the background solution procedure for trajectory optimization.

CHAPTER III

PRELIMINARY STUDIES

3.1 Obstacle Avoidance Framework Architecture

The base architecture of trajectory planning of this thesis is the *two-layer architecture* illustrated in Figure 7. The architecture consists of a receding horizon (RH) optimal trajectory generator and the vehicle controller. The RH trajectory generator repeatedly finds a feasible open-loop optimal trajectory by solving a finite-horizon constrained optimal control problem starting from the current state, and the vehicle controller plays the role of trajectory follower and vehicle stabilizer.



Remote Instruction
TRAJECTORY GENERATOR
ADAPTIVE FLIGHT CONTROLLER
MISSION DATA
Waypoints Instructions
BASE TRAJECTORY PLANNER
Reference Trajectory Commands
OUTER LOOP (Trajectory Tracking)
attitude, attitude rate commands
INNER LOOP (Stabilization)
actuator commands
actuators
Engine
VEHICLE
OBSTACLE DETECTION SENSOR
Obstacle geometry
REAL-TIME OPTIMAL TRAJECTORY GENERATOR
optimal Trajectory
Obstacle
attitude, actuator, throttle feedbacks
position, velocity, acceleration feedbacks

Figure 7: Two-layer architecture of the framework

The vehicle controller considered in this thesis is the adaptive nonlinear controller composed of the outer-loop and the inner-loop. The outer-loop receives the command trajectory from the trajectory generator and produces the command attitude to track the reference trajectory, and the inner-loop controls the vehicle attitude and stabilizes the vehicle. Even though the vehicle dynamics has a certain amount of uncertainty, the adaptive controller loops compensate for the errors from the uncertainty with an adaptive neural network (ANN). Details of the controller design is described in [62]. The vehicle controller tracks the reference trajectory and maintains the stability despite the existence of measurement noise, unmodeled dynamics, uncertainties and disturbances, which cannot be taken into account in the trajectory generator. This is a typical advantage of the two-layer architecture that guarantees the system follows a feasible trajectory along which the system can be stabilized.

Many studies on MPC or RHC for trajectory planning have targeted combining the trajectory generator with the stabilizing controller into a single trajectory planner. In this approach, the trajectory planner has the role of optimal controller that computes the control command to the vehicle actuator. However, although the stability of the RHC scheme can be guaranteed by the use of a special cost function within a certain finite horizon [60], in general, the corresponding optimization problem may not be computationally tractable if it has to account for all the realistic state and input constraints, unmodeled dynamics, as well as dynamically changing environments. Thus, an increase of complexity in the optimization problem may cause inadmissible delays in updating the command or the failure to find feasible solutions, which might result in a serious loss of stability. On the contrary, in the two-layer architecture, it is true that only suboptimal solutions can be attainable, as the trajectory optimizer cannot take into account the complex dynamics of the vehicle and actuators as well as internal nonlinear constraints. However, if done properly, only a small loss in performance will be carried whereas the stability will be maintained.

As Figure 7 depicts, the optimal trajectory generator is interfaced with the actual sensor and is integrated with the existing base trajectory planner in parallel. The base trajectory planner provides the waypoint navigation functionality for normal flight operation. Switching between normal waypoint navigation and obstacle avoidance can be automatic or manually selected by the operator in the ground station. In fact, the optimal trajectory planner can generate the command to the next waypoint at the current position, so, it can also provide a limited functionality of waypoint navigation without smooth transition between waypoints.

Several studies in the past on obstacle avoidance in uncertain environments have used obstacle detection sensors. Current state-of-the-art imaging or ranging sensors such as single camera [132], stereo cameras, scanning laser range finder (laser scanner) [44], and LIDAR [133] have been commonly chosen mostly for reactive obstacle avoidance. Vision sensors can provide wide and long field of view of the obstacle field with relatively accurate measurement on obstacle geometry, but they should accompany the image processing algorithms, requiring some amounts of computational power and time, and the size of detectable objects and the measurement accuracy of camera-based systems decrease rapidly with distance. A camera may be used to detect a tree but an individual branch of a tree or a telephone wire might be virtually invisible from a distance, and cameras are absolutely dependent on ambient lighting, so they can be easily influenced by changes in flight conditions such as time of day and weather.

On the other hand, radiation type sensors such as laser range finders and LIDARs demand less computational power and have fast acquisition capability. Small obstacles such as wires and poles can be better detected by this type of sensor and the measurement accuracy is almost constant within the detection range. However, laser sensors also have some disadvantages. For instance, specular reflection of the laser beam causes measurement errors, and a glossy finish on a car or a puddle of water remains invisible at shallow angles. In addition, the reflected beam may bounce off another obstacle and return to the sensor, giving a false measurement that makes the obstacle seem to be farther away than it is [113]. To make matters worse, those sensors can be blinded by the sun. Laser type sensors mountable on small UAVs have a limited detection range of less than a thousand feet, and only the exposed surfaces within the scanning region can be detected as a point cloud. Thus it requires another processing algorithm and the accumulation of the point cloud data to construct the complete geometry of the obstacle. In this thesis, a 2D scanning LIDAR fitted to the vehicle to measure the vertical profile of obstacles and terrain was used, as shown in Figure 8. So, in order to build a 3D geometry of the obstacle field, sinusoidal yaw attitude was commanded to the vehicle during flight.

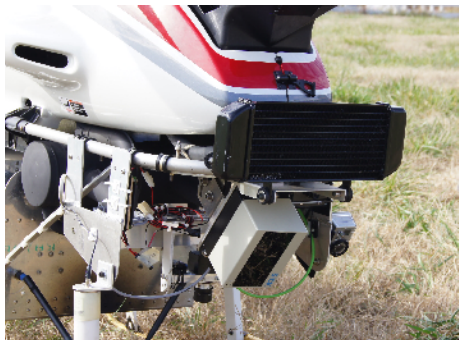


Figure 8: A LIDAR (Sick LD-MRS HD) fitted to the Geogia Tech UAV test-bed to scan vertically.

3.2 Time-optimal Obstacle Avoidance

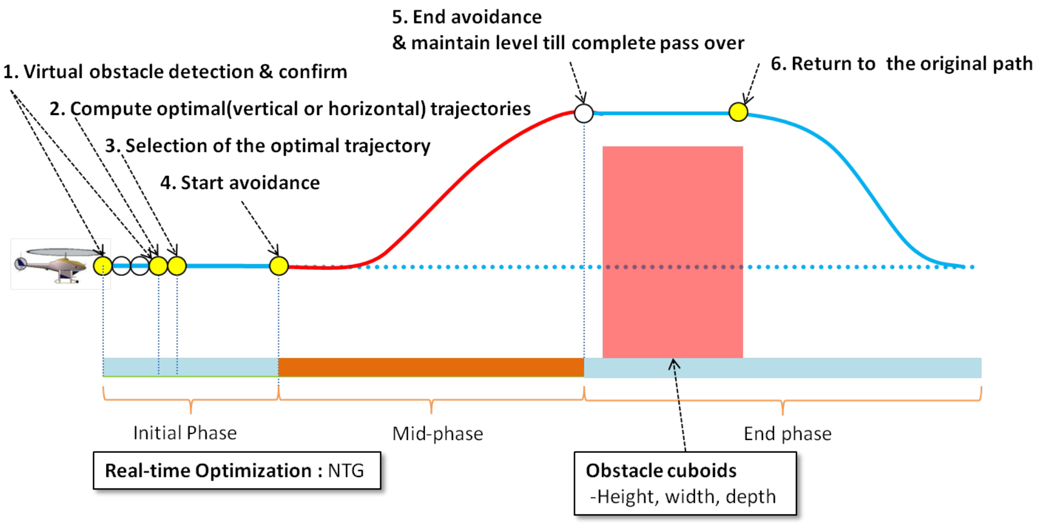

5. End avoidance
& maintain level till complete pass over
1. Virtual obstacle detection & confirm
6. Return to the original path
2. Compute optimal(vertical or horizontal) trajectories
3. Selection of the optimal trajectory
4. Start avoidance
Initial Phase
Mid-phase
End phase
Real-time Optimization : NTG
Obstacle cuboids
-Height, width, depth

Figure 9: 2D time-optimal obstacle avoidance scheme

Before the integrated framework was fully developed, the optimal trajectory generation scheme as illustrated in Figure 9 was implemented in the simulation software without the actual sensor interface, then it was ported to the onboard software of the Georgia Tech UAV test-bed. Details regarding the software implementation are presented in a later chapter. Because it had sensor interface, the trajectory generator used an obstacle cuboid given by height, width, and depth, which was transmitted to the framework externally during flight.

Figure 10 given below shows the guidance coordinate frame considered in the problem formulation. It is an inertial frame of which the x-axis is parallel to the line connecting the passed and the next waypoint, and the origin is fixed to the inertial frame. The coordinate switches subsequently whenever the target waypoint is reached. The position $\vec{p}$, the velocity $\vec{v}$, and the acceleration $\vec{a}$ vector are defined in this frame below.

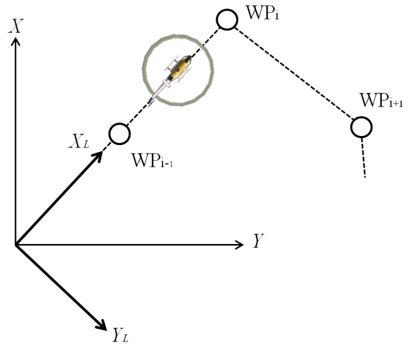


Figure 10: Coordinate frames defined in the problem formulation

3.2.1 Problem Formulation

Time-optimal trajectory has been traditionally chosen for the problem of optimal obstacle avoidance [94, 126] because of the practical consideration that a time-optimal solution can provide a minimized duration of avoidance or a minimum exposure to threats in hostile environments.

A time-optimal problem is formulated to minimize the final time of maneuver:

$$J = \int_0^{t_f} 1dt = t_f \tag{68}$$

and, focusing mainly on the generation of a trajectory, it is relevant to assume that the trajectory is subjected to the kinematics and simple 1st order model of vehicle dynamics with time-constant including the controller delay.

$$\vec{v} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{69}$$

$$\vec{a} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{70}$$

$$\dot{\vec{a}} = \begin{bmatrix} \dot{a_x} \\ \dot{a_y} \\ \dot{a_z} \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_x}(-a_x + a_{x_c}) \\ \frac{1}{\tau_y}(-a_y + a_{y_c}) \\ \frac{1}{\tau_z}(-a_z + a_{z_c}) \end{bmatrix} \tag{71}$$

where, $(x,\ y,\ z)$ is the position vector with respect to the guidance frame, $(u,\ v,\ w)$ is the velocity, $(a_x,\ a_y,\ a_z)$ is the acceleration, $(a_{x_c},\ a_{y_c},\ a_{z_c})$ are the command accelerations, and τ_x, τ_y, τ_z are the time constants of respective accelerations.

The system of Equations (69) through (71) is a typical flat system having the position (x, y, z) and flat outputs as $z_1 = x$, $z_2 = y$, and $z_3 = z$. Each flat output is parametrized with a B-spline of the order of $k = 6$, multiplicity $m = 4$, smoothness $s = 2$, and the number of intervals $l = 5$. From Equation (53), the selected B-spline parameters make 14 control points for each B-spline of flat output, forming a converted NLP with the vector of 42 unknowns as given by Equation (66). In the selection of B-spline parameters for any flat output, a careful trade-off between the curve complexity required in the problem and the computation time for optimization is required. An increase in the number of B-spline coefficients can better approximate a complex variation of a flat output, but it may result in an increase of computation time to an unacceptable level for real time optimization.

In fact, Equations (69) and (70) are enough for the trajectory optimization of small high agility UAVs equipped with electric power of which the dynamics is sufficiently fast to be ignored. However, any desired trajectory has to be realized by the vehicle autopilot and the vehicle. The inclusion of the accurate vehicle and the autopilot model would produce solutions more dynamically feasible and accurate, but the dimension of the dynamics could easily become high-order, and it can increase the computational complexity of the solution, causing practical real-time computation issues. Therefore, for a simplification in representing the vehicle dynamics for the optimization procedure, the 1st-order model given by Equation (71) has been used from our past studies on time-optimal avoidance [68, 94]

The time-optimal obstacle avoidance is focused on finding the optimal avoidance trajectory for a given obstacle cuboid based on the strategy: *the vehicle keeps the current path and approach speed until the last moment, when it should initiate a horizontally non-accelerating avoidance maneuver.* Thus, this avoidance strategy is designed to get the minimum mission speed deviation as well as minimum duration to maneuver and on the other hand to maximize the non-avoidance mission path.

3.2.1.1 Initial Constraints

Provided that the vehicle has the constant horizontal speed (u_{cmd}), and from the strategy described above, the starting point of avoidance can be freed in front of the vehicle and the zero-acceleration condition can be taken as the initial constraint, as given by:

$$\begin{aligned}\vec{p}(0) &= [free,\ y_0,\ z_0]^T \\ \vec{v}(0) &= [u_{cmd},\ 0,\ 0]^T \\ \vec{a}(0) &= [0,\ 0,\ 0]^T\end{aligned} \tag{72}$$

So, now the optimization problem becomes a problem of finding the optimal avoidance initiation position as well as an avoidance trajectory.

3.2.1.2 Terminal Constraints

Obviously, the final time should be free and the vehicle should be outside of the obstacle for avoidance. So, in the first preliminary study, as the obstacle was assumed to be a cuboid, an intermediate safe waypoint (x_{safe}, y_{safe}, z_{safe}) was selected outside the cuboid as the terminal position of the avoidance that the vehicle should fly through at the end of the time-optimal maneuver. In addition to that, a level flight condition with zero vertical speed and a zero-acceleration condition were selected for the smooth transition to the returning path after avoidance.

$$\begin{aligned}
&t_f = free \\
&\vec{p}(t_f) = [x(t_f), y(t_f), z(t_f)]^T = [x_{safe}, y_{safe}, z_{safe}]^T \\
&\vec{v}(t_f) = [u(t_f), v(t_f), w(t_f)]^T = [u_f,\ 0,\ 0]^T \\
&\vec{a}(t_f) = [a_x(t_f), a_y(t_f), a_z(t_f)]^T = [0,\ 0,\ 0]^T
\end{aligned} \tag{73}$$

where the safe waypoint can be determined from the obstacle rectangular geometry and the safety clearance Δr_s, and the final speed u_f is bounded as given by

$$\begin{aligned}
x_{safe} &= \min x_{ob} - \Delta r_s \\
y_{safe} &= \begin{cases} \min y_{ob} - \Delta r_s,\ \text{if } |\min y_{ob} - y_0| \leq |\max y_{ob} - y_0| \\ y_0,\ \text{if } z_{safe} \neq z_0 \\ \max y_{ob} + \Delta r_s,\ \text{if } |\min y_{ob} - y_0| \geq |\max y_{ob} - y_0| \end{cases} \\
z_{safe} &= \begin{cases} \max z_{ob} + \Delta r_s,\ \text{if } y_{safe} = y_0 \\ z_0,\ \text{if } y_{safe} \neq y_0 \end{cases} \\
u_f &= [0, u_{cmd}]
\end{aligned} \tag{74}$$

3.2.1.3 Path Constraints

At any instance of the avoidance maneuver, the accelerations and velocities should be maintained within allowable boundaries by the performance limit or the envelope limit like the V-n diagram. This thesis considered that the acceleration was constrained by the ellipsoidal equation of total acceleration as given by

$$0 \leq \left(\frac{a_x}{a_{x_{max}}}\right)^2 + \left(\frac{a_y}{a_{y_{max}}}\right)^2 + \left(\frac{a_z}{a_{z_{max}}}\right)^2 \leq 1 \tag{75}$$

where the maximum values of each acceleration component can be determined from the vehicle performance limit or the envelope limit.

The velocity of the vehicle can be bounded separately along with the total velocity

$$\begin{aligned} 0 &\le u \le u_U \\ v_L &\le v \le v_U \\ w_L &\le w \le w_U \\ V_{t_L} &\le \sqrt{u^2 + v^2 + w^2} \le V_{t_U} \end{aligned} \tag{76}$$

where the longitudinal speed limit u_U is set to the desired command speed u_{cmd}, and lateral speed limits v_L and v_U are set to zero for pure vertical avoidance.

The actual climb rate limit w_U is closely related to the engine power. Theoretically, the climb rate limit can be determined from the maximum excess power but unfortunately most small rotary-wing UAVs usually are not equipped with any onboard device to measure the engine power directly, so the power limit should be estimated indirectly from the other parameters available. Using an inaccurate estimate of the climb rate limit, especially if the value is overestimated, may result in an unsafe situation in the avoidance where the maximum climb rate is required, as in the case of detecting a large obstacle at a short distance. The worst case is that the vehicle cannot follow the optimal command trajectory, resulting in a collision because the command trajectory is actually infeasible to the vehicle performance due to the overestimation of performance limit. This problem occurred during the first flight test. The flight test result is presented in a later section.

Another point of the formulation is that there is no path constraint related with the clearance of the obstacle shape, so the computational load is actually less than it would be if a realistic obstacle shape were used. The optimal problem can be solved in a relatively short time, and by taking advantage of real-time optimization techniques, the problem can be solved quickly for multiple times at the moment that the obstacle geometry is provided. The fast computation of vertical and horizontal optimal trajectories are useful in deciding the optimal direction for avoidance as well as the avoidance trajectory. Figure 11 depicts this concept.

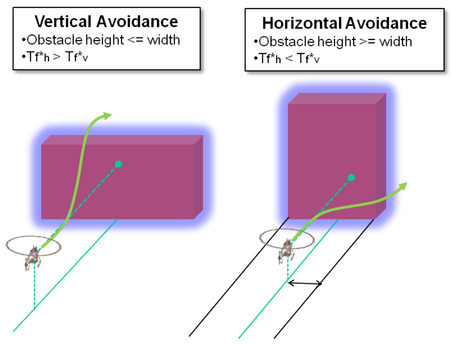


Figure 11: Selection of avoidance direction

3.2.2 Time-optimal Trajectory Using Pseudo-spectral Method

Time-optimal solutions for vertical obstacle avoidance had been explored by different real-time optimization methods before the development of the obstacle avoidance framework to integrate the real-time optimal solver. The pseudo-spectral method and NTG were investigated in such alternative studies. As a pseudo-spectral method, an open source software library, PSOPT [4], was used to solve the same problem solved by NTG, and the results were compared for evaluation. PSOPT is written in a C++ and can be used to solve general optimal control problems of continuous nonlinear systems with constraints. It has C++ programming interface that can facilitate the OCP problem formulation in programming, automatic differentiation to get Jacobian and Hessian matrices, automatic identification of sparsity, automatic spectral node refinement, and selectable NLP solvers, SNOPT [46] and IPOPT [70]. The PSOPT is capable of solving a more complex optimization problem that has complex nonlinear dynamic equations, multiphase problems, general nonlinear constraints even with interior point constraints, and free or fixed initial and terminal times, even with differential equations with delayed variables.

Figure 12 is an example of the time-optimal trajectory obtained from both NTG

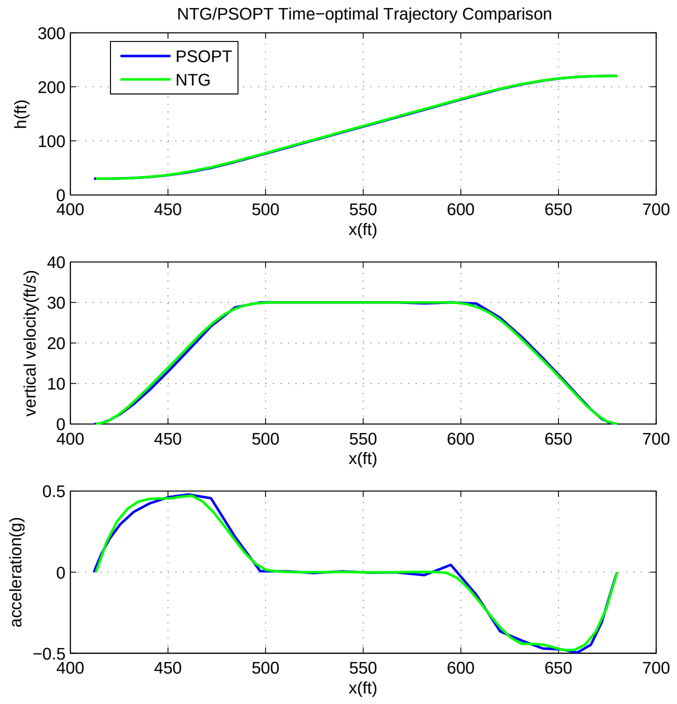


Figure 12: Time-optimal avoidance trajectories from NTG and PSOPT: Obstacle is located at $x = 700ft$, height is $200ft$; safe waypoint is set to ($680ft$, $0.0ft$, $220ft$); the climb rate limit is set to $30ft/s$; and vertical acceleration boundary is $-0.5g \leq a_z \leq 0.5g$.

and PSOPT, that shows the optimal trajectories obtained are almost the same using two different methods. Both solvers compute the suboptimal solutions so the results from both solvers might be slightly different because of the difference in the detailed approach to transcription of OCP to NLP and the NLP solvers. In the specific example shown in Figure 12, the optimal terminal times given by both solvers are almost same, $t_f = 8.89418$ by PSOPT and $t_f = 8.894$ by NTG. However, in general, NTG takes less computation time for the same problem compared to the PSOPT; on the other hand, PSOPT made it easy to program the computation problem and showed robust convergences of solutions despite rough initial guesses. The trade-off between robustness to the initial guess and the computation time should be considered when choosing the primary real-time optimization solver. Especially for obstacle avoidance, fast computation time is critical when the vehicle motion and flight environment are changing rapidly and unexpectedly. For this reason, NTG was finally selected as the optimal solver for the framework.

3.2.3 Time-optimal Avoidance of Multiple Obstacles

The problem formulation of the time-optimal avoidance for a single obstacle does not have any constraint or cost to take into account the shape of obstacle, and actually the optimal problem only focuses on the trajectory before the obstacle. In case of multiple obstacle avoidance, the previous approach can be applicable for the sequential or phase-wise optimization by segmenting the overall avoidance problem into small problems between sequences of obstacles, but this approach can end up with a non-optimal solution for the entire obstacle avoidance. A more appropriate approach for multiple obstacles is to formulate the obstacle geometry as path constraints, and one way to represent a simplified obstacle shape such as circle or rectangle is using the *p-norm* relation

$$h_i(x,y,z) = \left[\left(\frac{x - x_{c_i}}{a}\right)^p + \left(\frac{y - y_{c_i}}{b}\right)^p + \left(\frac{z}{c}\right)^p\right]^{\frac{1}{p}} - 1 \geq 0 \tag{77}$$

where $(x_{c_i},\, y_{c_i})$ is the center position of the obstacle, a is the half length plus clearance, b is the half width plus clearance, c is the height of the obstacle plus clearance. Figure 13 represents the shape of the unit *p-norms*. In fact, using Equation (77)

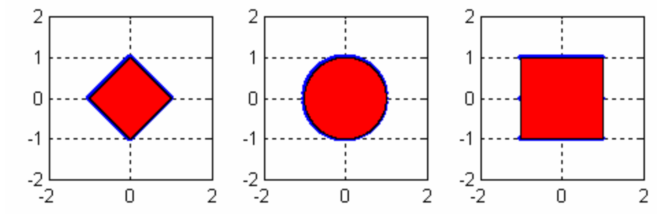
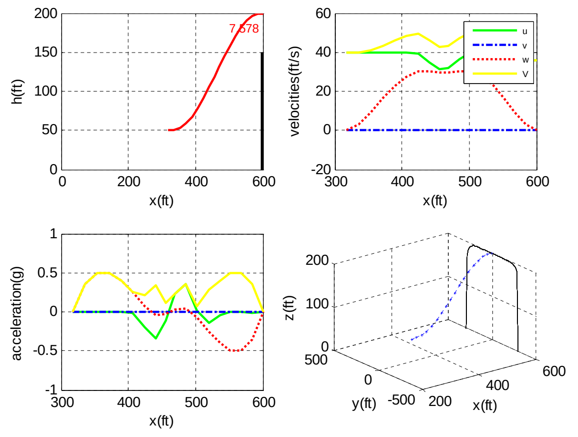
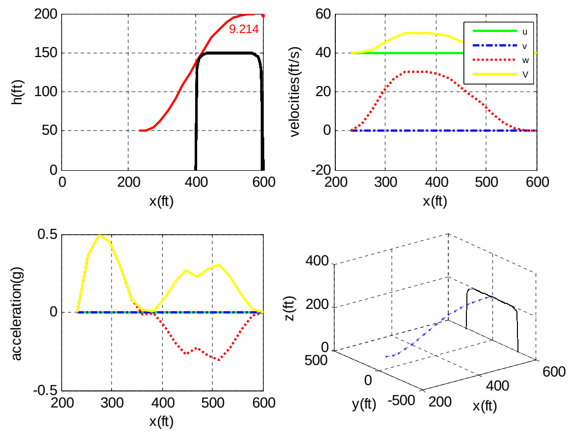


Figure 13: Unit P-norms: p=1, 2, and ∞

to represent the perfect rectangular shape is problematic, for it requires $p \to \infty$ for rectangles that can easily cause computational difficulties in numerical iteration of optimization. For this reason, this thesis used a relatively small number, p=*8*, to roughly approximate the obstacle cuboid. Figures 14 and 15 show the example solutions of time-optimal trajectories in the presence of a middle obstacle with different locations and sizes, showing the effect of the middle obstacle on the variations of the time-optimal solution.


(a) no middle obstacle ($t_f^* = 7.578$)


(b) middle obstacle $h = 50ft$ at $400ft$ ($t_f^* = 9.214$)

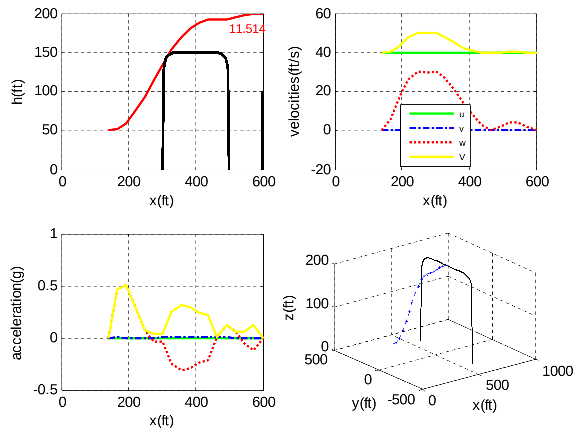
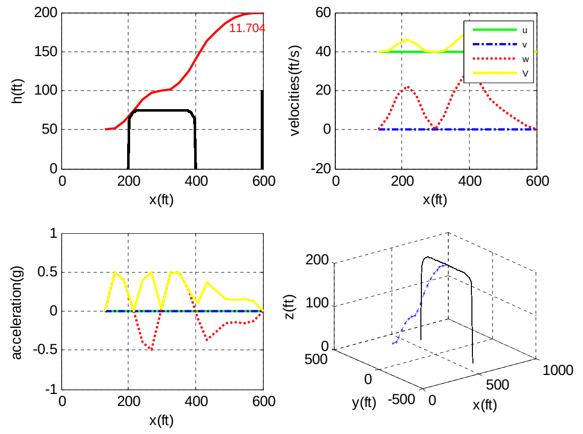
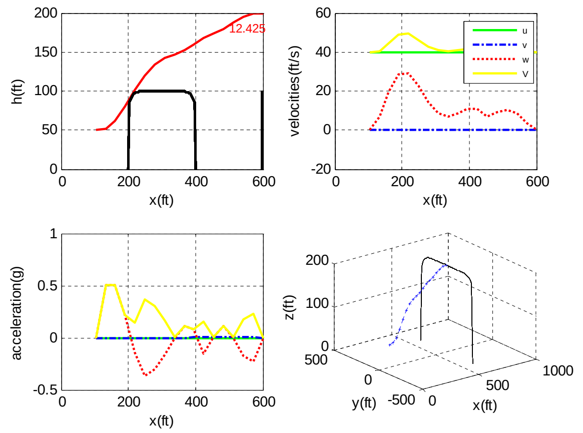

(c) middle obstacle $h = 150ft$ at $300ft$ ($t_f^* = 11.514$)

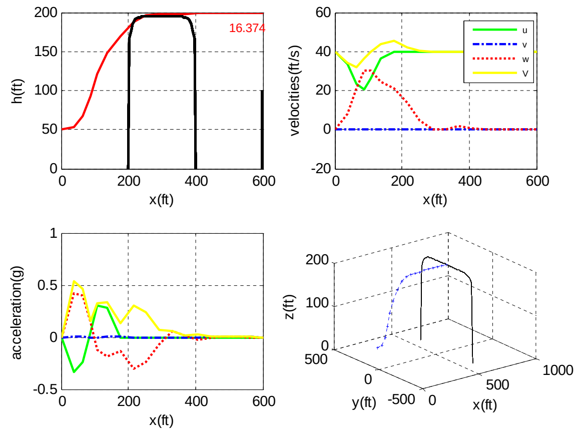
Figure 14: Time-Optimal avoidance for multiple obstacles: the variation of time-optimal trajectory by middle obstacle; main target is located at $600ft$ and height is $200ft$ (including clearance); initial vehicle location is ($0ft$,$0ft$,$50ft$); initial speed is $40ft/s$; climb rate and total velocity are limited to $30ft/sc$ and $50ft/s$, respectively.



(a) no middle obstacle $h = 75ft$ at $200ft$ ($t_f^* = 11.704$)



(b) middle obstacle $h = 100ft$ at $200ft$ ($t_f^* = 12.425$)



(c) middle obstacle $h = 200ft$ at $200ft$ ($t_f^* = 16.374$)

Figure 15: Time-optimal avoidance for multiple obstacles: the variation of time-optimal trajectory by the change of middle obstacle height; initial and constraints are the same as in Figure 14.

3.2.4 Time-optimal Avoidance of Terminal Manifold

In practical situations, it is better to avoid a tall building by flying around rather than flying above it. A generic approach for such a situation is to consider the exact shape of an obstacle but this comes with the cost of increased mathematical and computational complexity. Assuming that the sensor can detect any arbitrary shape of an obstacle within its measurement range, the optimal trajectory should skim the area occupied by obstacles in a plane perpendicular to the flight velocity at the final time t_f. This arbitrary shaped area occupied by single or a group of obstacles is defined as the *terminal manifold*. Figure 16 shows the concept of this avoidance problem.

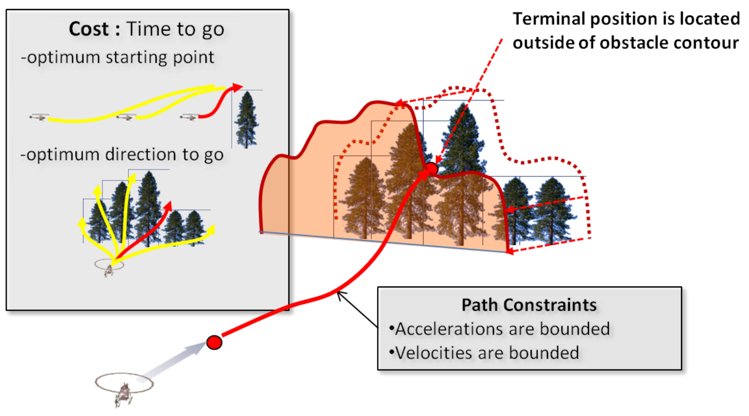
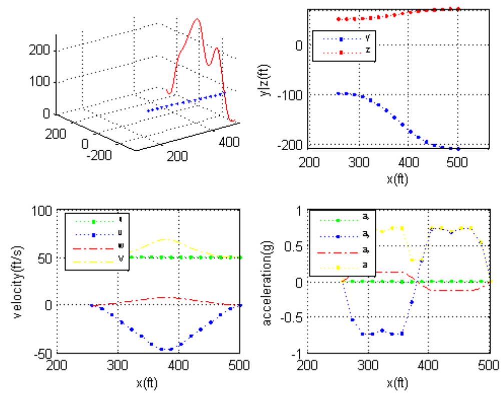


Figure 16: Concept of time-optimal avoidance for the terminal manifold

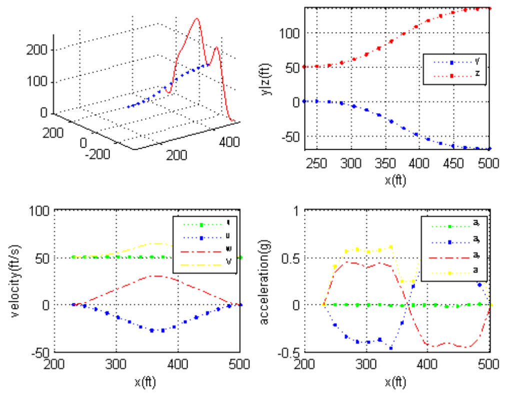
In order to solve the time-optimal trajectory for a terminal manifold, the problem needs an extra constraint to the original formulation consisted of Equations (68) through (76), replacing the terminal constraints of Equations (73) and (74).

$$\begin{aligned} x_{safe} &= x_{mf} \\ y_{safe} &= y_{mf} = free \\ z_{safe} &\geq \mathcal{M}(y_{mf}) \end{aligned} \tag{78}$$

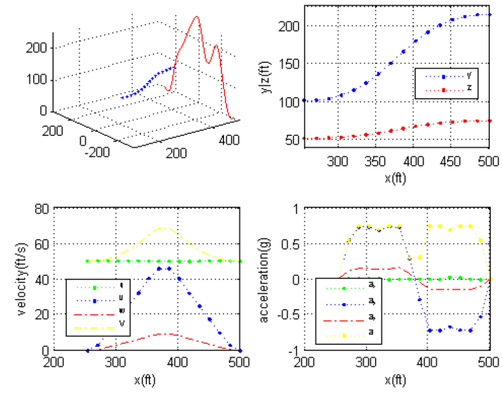
where x_{mf} is the longitudinal position, y_{mf} is the arbitrary lateral position at x_{mf}, and $\mathcal{M}(y_{mf})$ is the height of the manifold at y_{mf}. The shape of the manifold is considered to be arbitrarily given by the discrete points on the manifold. For this reason, the manifold is interpolated with the cubic spline interpolation which enables the continuous approximation of arbitrary shape given by discrete points as well as the derivatives at any point on the manifold outline.



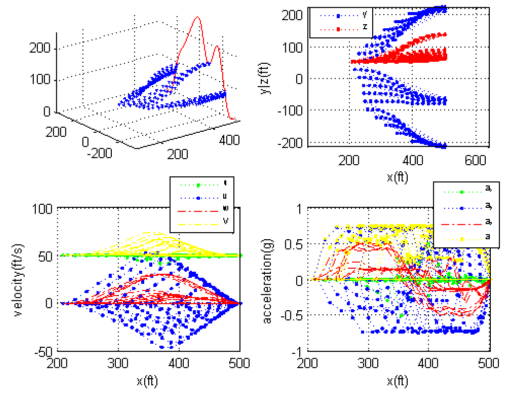
(a) $y_0 = -100ft$



(b) $y_0 = 0ft$



(c) $y_0 = 100ft$

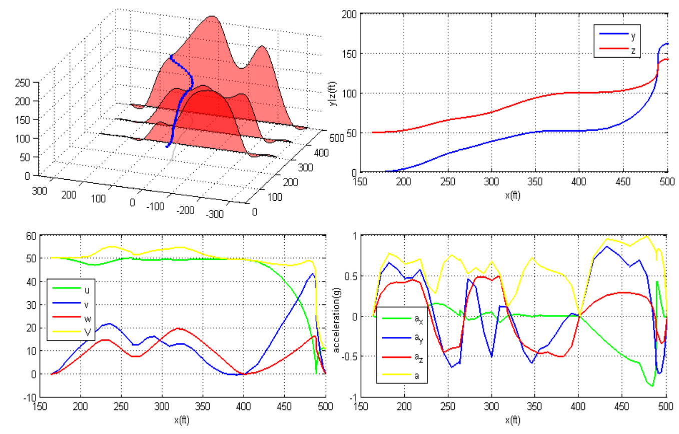


(d) $y_0 = -200 \sim 200ft$

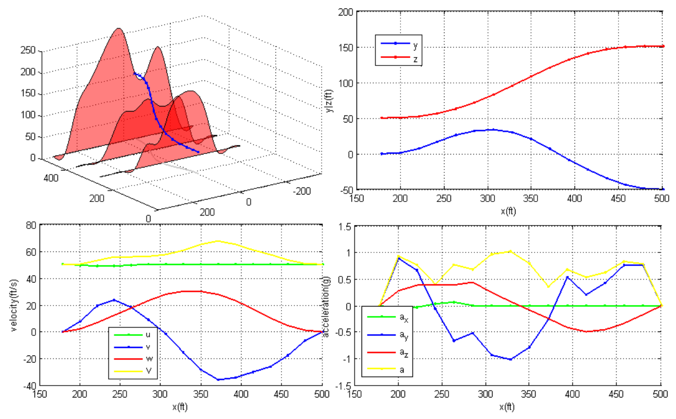
Figure 17: Time-optimal avoidance for arbitrary terminal manifold: The manifold is located at $500ft$, initial vehicle longitudinal location is at $0ft$, height is $50ft$, initial velocity is $50ft/s$.

Figure 17 shows the time-optimal trajectories for a given terminal manifold depending on the initial lateral position of the vehicle. The approach could find relevant optimal trajectories; when the vehicle is located initially at side of the manifold, Figure 17(a) and (c), the solution trajectory seeks the nearest points on the manifold, whereas when the vehicle is located relatively at the center, the solution finds the trajectories to the valley in the middle of the manifold, see Figure17(b).

The terminal manifold approach was later extended to the problem of multiple layers of the manifold, assuming the slice of arbitrary obstacle geometry can be provided during flight. Figure 18(a) shows the time-optimal solution for the three layers of manifolds detected subsequently at the end of the previous layer, and Figure 18(b) represents the case that the same layers are detected at once initially. A comparison of both results reveals the fundamental aspect of the local trajectory optimization: *local optimum solutions do not guarantee the global optimality.*



(a) sequential detection ($t_f^* = 10.54$)



(b) simultaneous detection ($t_f^* = 6.42$)

Figure 18: Time-optimal avoidance for arbitrary multiple manifolds

3.2.5 Flight Test Evaluation of The Real-Time Optimizer

The time-optimal avoidance approach was implemented in the simulation tool, the Georgia Tech UAV Simulation Tool (GUST) and the onboard software of the GT-Max, the Georgia Tech rotary-wing UAV test-bed, and the in-flight real-time optimal trajectory generation capability using virtual obstacle data was tested in a flight test. Details of the GUST and GTMax system are presented in [64, 65]. Figure 19 shows the implementation of the framework for the first flight test. The framework has been improved and the software implementation and integration of the final framework are presented in Chapter 5. As depicted in Figure 19, the optimal avoidance trajectory generation module, named INTOPTOA, was embedded in the Onboard2 computer of the GTMax without the sensor.

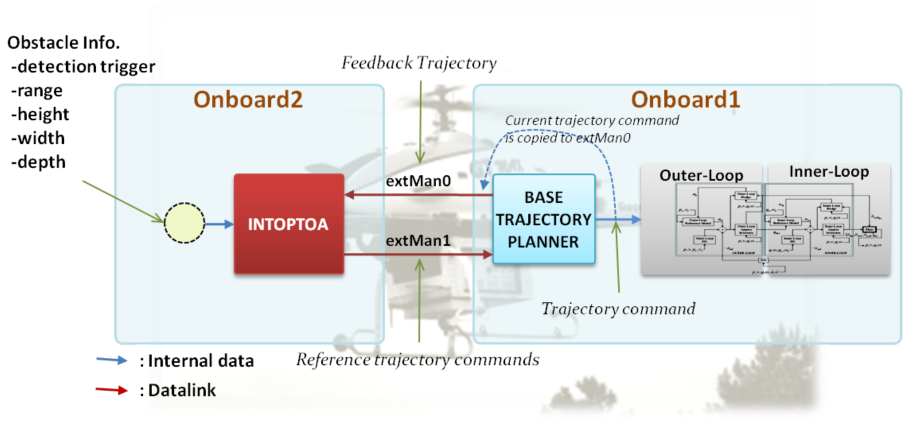


Figure 19: Software implementation for the first flight test

The avoidance maneuver was conducted for the virtual obstacles which were prepared on the ground and transmitted to the vehicle while it was flying over the preplanned test trajectory mainly for safety reasons. The obstacle data includes the detection trigger, range, height, width, and depth of the obstacle. The detailed values of the data are presented in Figure 20, and the limit parameters, shown in Table 1, were chosen to be smaller than those expected in the simulation model to lessen the aggressiveness of the avoidance maneuver. At the flight test, the clearance distance was set to $10ft$ in the vertical plane and the horizontal clearance was determined automatically by the available acceleration, the flight speed, and the marginal computation delays, approximately set to $100ft$ during flight.

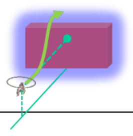
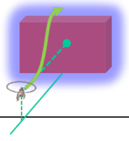
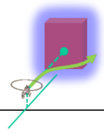
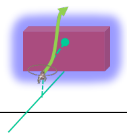
Obstacle Cases	Case0	Case1	Case2	Case3
Detected Range (ft)	600	600	600	300
Detected Height (ft)	350	400	350	350
Detected Width (ft)	400	500	150	400
	Vertical	Vertical	Horizontal	Short Detection Vertical
Avoidance				

Figure 20: Time-optimal flight test cases and avoidance conditions

Table 1: Limit values for the flight test

a_x, a_y	-0.3g $\sim$ 0.3g
a_z	-0.5g $\sim$ 0.5g
w	$30ft/s$ (for all case), 25, $20ft/s$ (for case 0)

Figures 21 through 26 show the flight test results for each of the cases in Figure 20, which show the position command and response, and the climb rate command and response. The commands (plotted in red) were computed in real time as expected in the simulation, but in most vertical avoidance, the responses (plotted in blue) did not follow the commands as expected in the simulations, except for the horizontal avoidance case (Figure 23) and the vertical avoidance with a climb rate limit of $20ft/s$ (Figure 26). In the most severe case of vertical avoidance, case 3 with the short detection range, the vehicle actually collided with the obstacle in virtual space, but when the climb rate limit was decreased to $20ft/s$, the vehicle could appropriately follow the command trajectory as expected in the simulation, as seen in Figure 26.

The flight test results were different from those expected in the simulation model (unlike the flight test results, the simulation indicated that the vehicle would follow the command trajectory well). However, compared to the vertical avoidance, as shown in Figure 23, the horizontal avoidance trajectory did not show a significant lag to the command trajectory. From these results, the actual maximum climb rate was around $20ft/s$, and the maximum available power in the simulation model needed to be modified for a more accurate simulation. In order to validate this fact, a simulation was done by changing the maximum engine power, and when the maximum engine power was decreased by about 25%, the simulation could produce a similar output to the flight test result as seen in Figure 27.

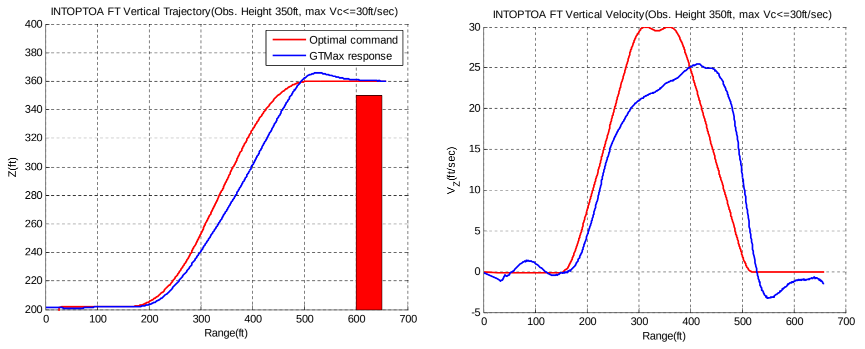


Figure 21: Time-optimal flight test case0

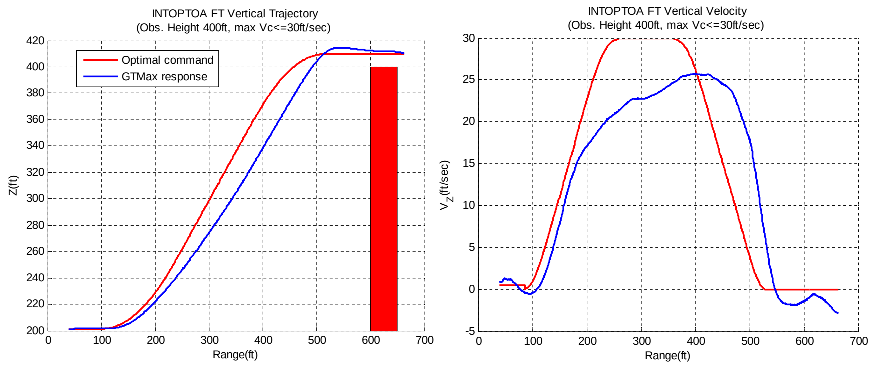


Figure 22: Time-optimal flight test case1

The results of the flight test cannot be mainly attributed to the decrease of the engine power since the variation of actual vehicle weight, the change in aerodynamic drag, the weather, and other unknown effects might also have contributed to the results. However, the flight test and the simulation results revealed that using the inaccurate envelope limit could endanger the vehicle during an avoidance maneuver. Therefore, it should be noted that the selection of the feasible envelop limit parameter and using an accurate value for that parameter are important for avoidance trajectory generation by the optimization.

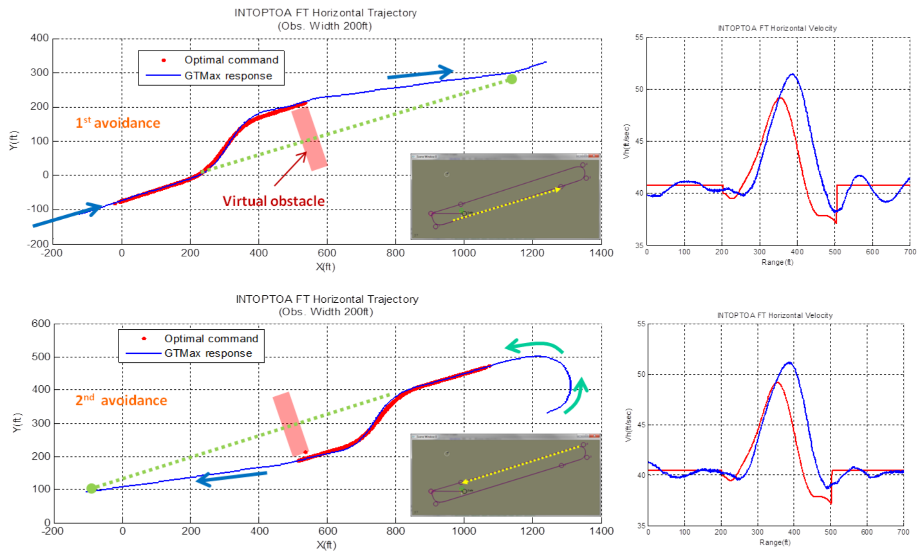


Figure 23: Time-optimal flight test case2

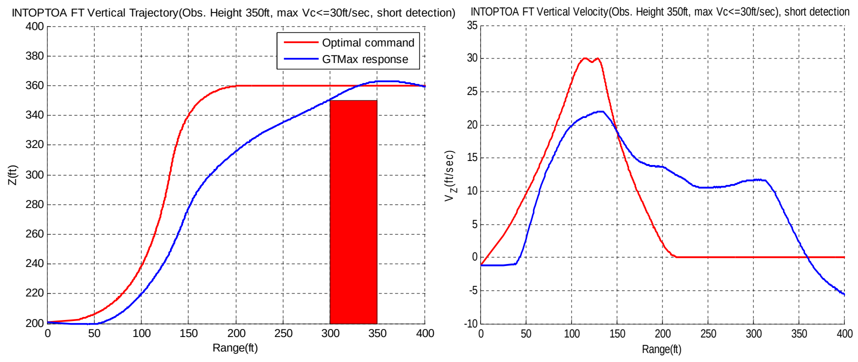


Figure 24: Time-optimal flight test case3

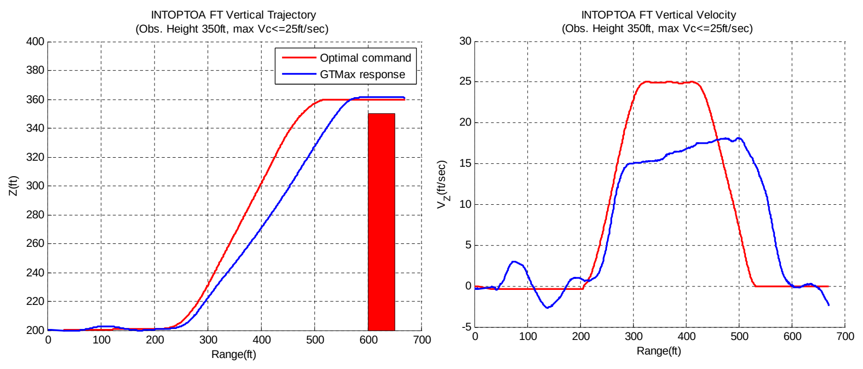


Figure 25: Time-optimal flight test case0 with climb rate limit of $25ft/s$

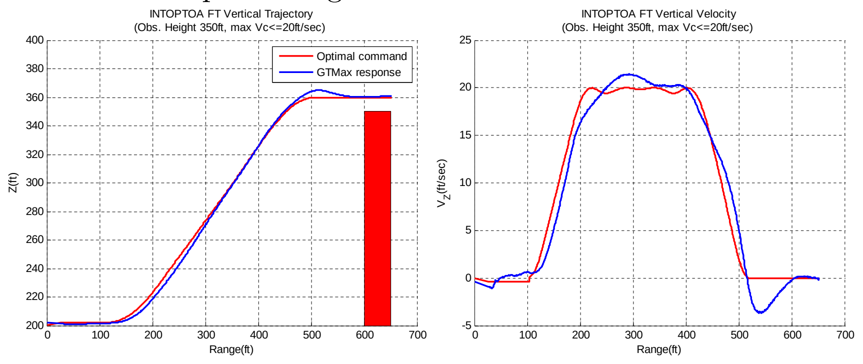


Figure 26: Time-optimal flight test case0 with climb rate limit of $20ft/s$

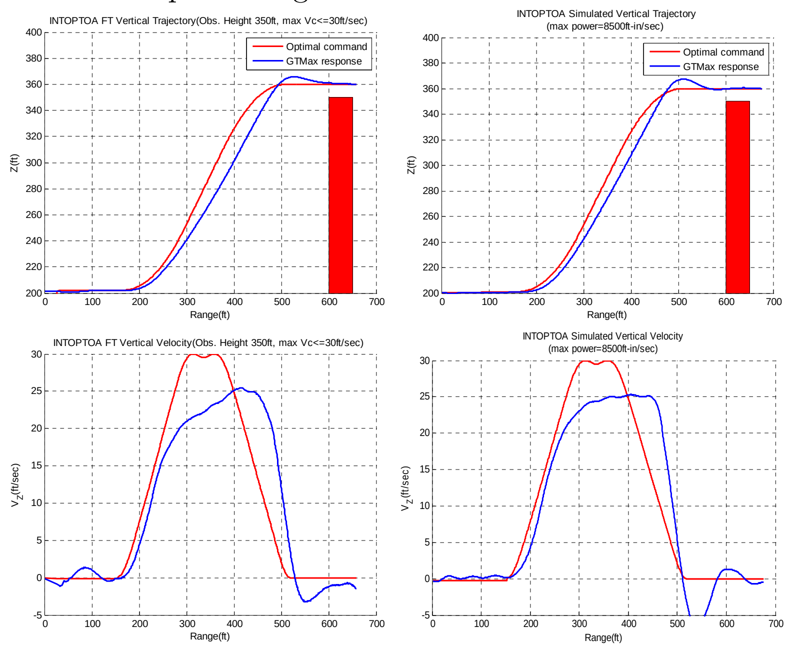


Figure 27: Comparison of time-optimal flight test case0 (left) to the simulation with decreased engine power (right).

3.3 Continuous Vertical Optimal Obstacle Avoidance

The time-optimal approach for avoidance was extended to the optimal avoidance with the real sensor interface. A LIDAR was selected to measure the obstacle and terrain profile in front of the vehicle within a finite range, so the continuous contour of the terrain and the obstacle was available for the optimization. However the key changes in the optimization scheme arose from the use of the sensor, *so that there was no longer complete knowledge of an obstacle on the infinite horizon of the obstacle field*, thus the scope of the optimization had to be limited to the sensor range. Practically, laser-type sensors mountable on small UAVs have a relatively small range, less than thousand feet, at most.

The finite range of the sensor required a new optimization approach to determine the optimal trajectory within the sensor range and to update the solution continuously, as the measured obstacle and terrain contour change as the vehicle advances. The *receding horizon trajectory optimization* scheme was first applied to this problem of continuous vertical optimal avoidance, and Figure 28 depicts this concept.

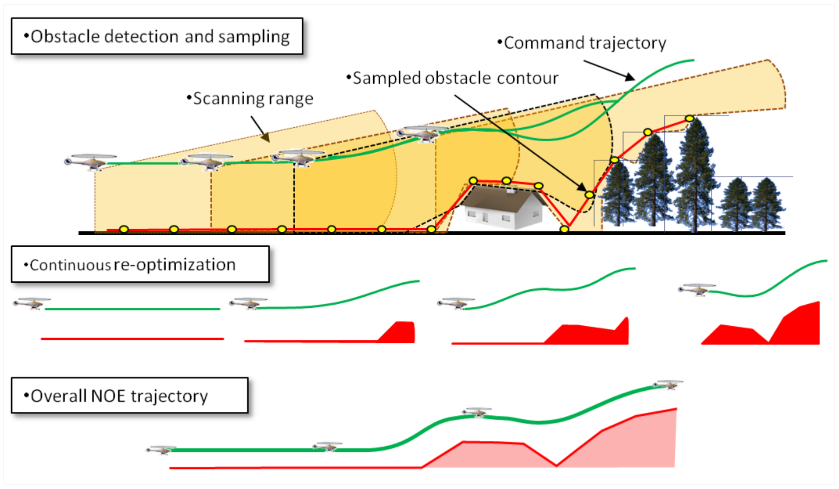

•Obstacle detection and sampling
•Command trajectory
•Sampled obstacle contour
•Scanning range
•Continuous re-optimization
•Overall NOE trajectory

Figure 28: Concept of receding horizon trajectory optimization for vertical obstacle avoidance

Along with the change in the optimization scheme, the objective of the trajectory planning was also changed from the time-optimal avoidance to the optimal terrain following (TF) and to nap of the earth (NOE) flight. In TF or NOE situations, as described in the introduction, minimizing the deviation from the terrain or the desired height have more significance than the time-optimal maneuver that may result in a large height deviation from the terrain. Figure 29 compares fundamental differences in the two optimal trajectory solutions over the same obstacle geometry.

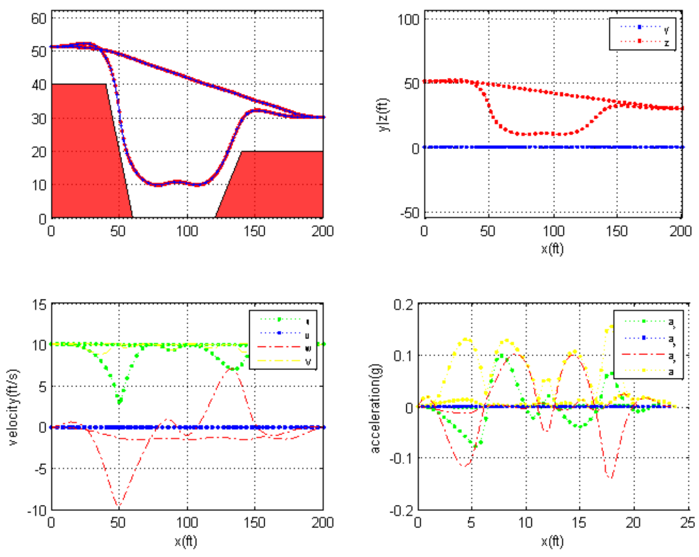


Figure 29: Comparison of example optimal trajectories: the time-optimal vs the minimum height deviation.

3.3.1 Problem Formulation

The trajectory optimization for minimizing the height deviation from the terrain contour can be formulated using the height difference as the term of cost

$$J = \int_0^{t_f} (z(t) - z_o(t))\, dt \tag{79}$$

where z_o is the height profile of the terrain or obstacle. The system dynamics given by Equations (69) through (71) can be used for this problem too, but other constraints need to be changed.

At first, the current vehicle position, velocity, and acceleration is set to the initial constraint instead of the free position and level flight condition in Equation (72)

$$\begin{aligned}\vec{p}(t_0) &= [x_0,\ y_0,\ z_0]^T \\ \vec{v}(t_0) &= [u_0,\ v_0,\ w_0]^T \\ \vec{a}(t_0) &= [a_{x_0},\ a_{y_0},\ a_{z_0}]^T\end{aligned} \tag{80}$$

and the terminal time, position, and velocity are free but the terminal acceleration is set to zero-acceleration.

$$\begin{aligned}t_f &= free \\ \vec{p}(t_f) &= [x_f,\ 0,\ z_f]^T,\ \ x_f \le R_{sensor},\ z_f \ge z_o(x_f) + r_s \\ \vec{v}(t_f) &= [u_f,\ 0,\ w_f]^T,\ \ 0 \le u_f \le u_U,\ w_L \le w_f \le w_U \\ \vec{a}(t_f) &= [0,\ 0,\ 0]^T\end{aligned} \tag{81}$$

where R_{sensor} is the maximum sensor range, and r_s is the clearance distance.

The acceleration and velocity should be confined within the boundary given in Equations (75) and (76) at all moments of the flight, and in addition, this problem formulation requires an extra path constraint that height should be constrained in order to give safety clearance r_s above the terrain or obstacle contour.

$$z(t) \ge \max(z_o(x(t)) + r_s, z_{min}) \tag{82}$$

where z_{min} is the minimum allowable height and the vertical contour of the terrain z_o is interpolated by the longitudinal position x from the points measured by the sensor.

3.3.2 Simulations

The simulation of the new vertical obstacle avoidance framework was done by renewed implementation of the problem formulation and the sensor model. Details on implementation of the simulation are presented later in Chapter 5. Figure 30 shows a captured scene in GUST during vertical obstacle avoidance. By using the modified framework, typical cases of the vertical avoidance were simulated to investigate the performance of the framework, and they are presented as Figures 31 through 33.

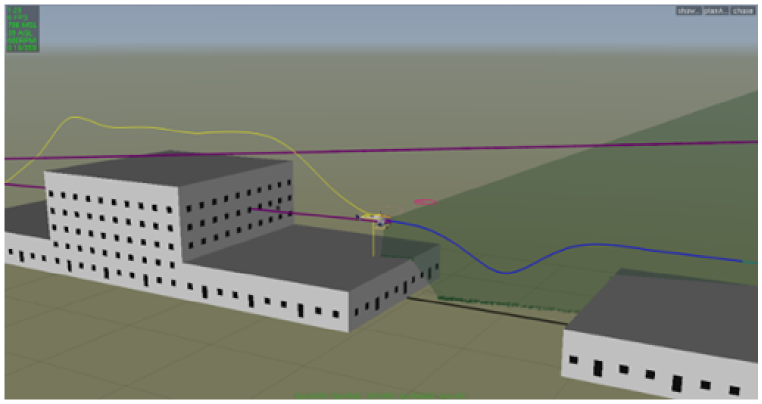


Figure 30: Real-time simulation scene in GUST for vertical optimal avoidance: Green region represents the sensor scan plane; Blue line is the real-time optimal command; Yellow line is the response trajectory.

Figure 31 is for the simulation with $10ft/s$ of command horizontal speed, Figure 32 is for $20ft/s$, and Figure 33 is for $30ft/s$, respectively. The height clearance is $20ft$, the maximum climb rate is $20ft/s$, and the maximum descent rate is $15ft/s$ for all cases. Figures 31 through 33 show that if the command horizontal speed is low, flight height tends to be close to the obstacle contour and the command trajectory can be followed with less perturbation.

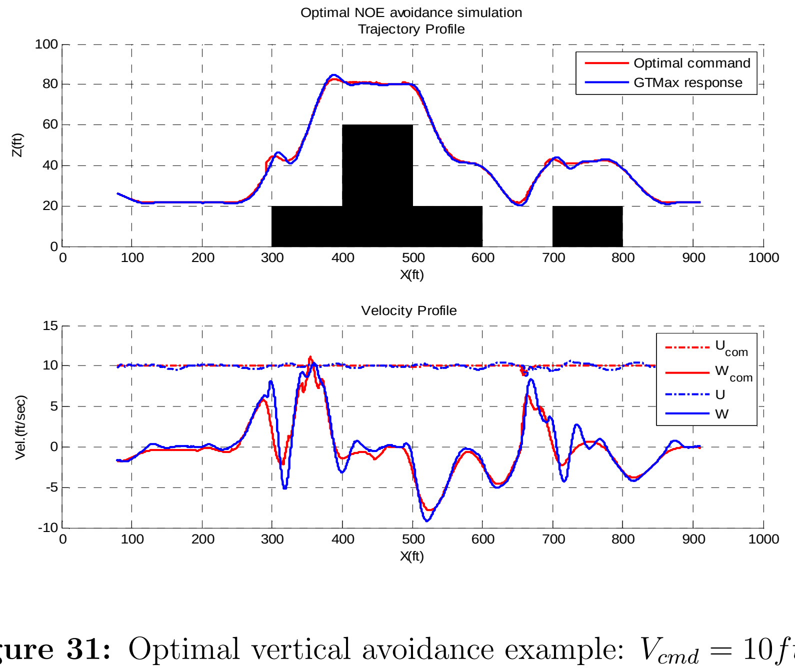


Figure 31: Optimal vertical avoidance example: $V_{cmd} = 10ft/s$.

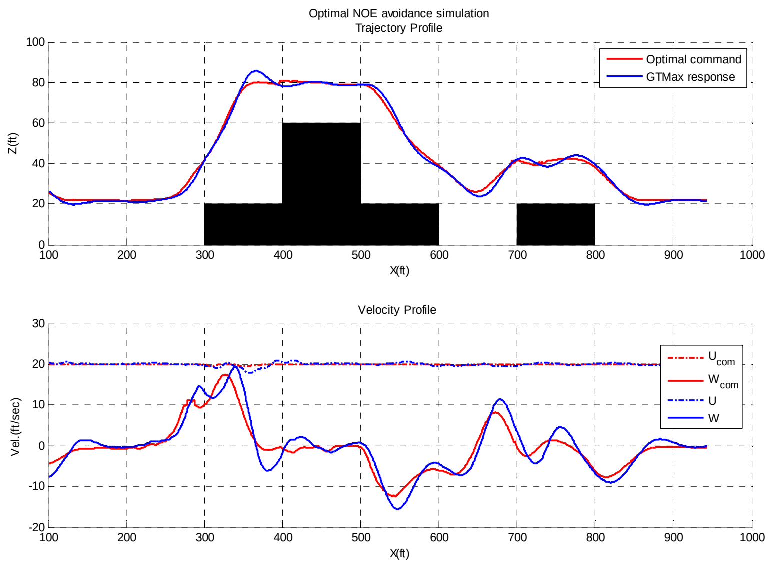


Figure 32: Optimal vertical avoidance example: $V_{cmd} = 20ft/s$.

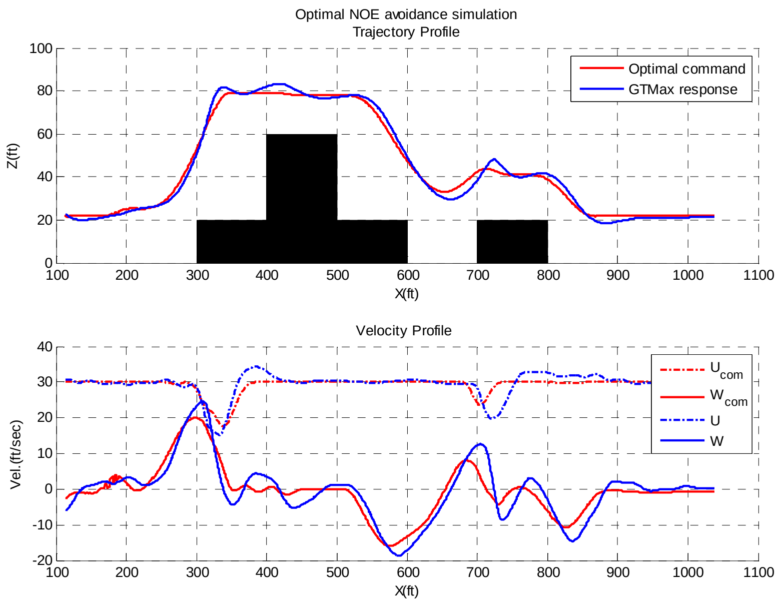


Figure 33: Optimal vertical avoidance example: $V_{cmd} = 30ft/s$.

Figure 34 shows an example simulation of sensor saturation, in which the sensor could not detect the actual height of the obstacle. In the figure, when the sensor is saturated, the optimal solution becomes a decrease in horizontal speed to nearly zero while the vertical speed increases to go over the obstacle, and after the sensor is recovered, horizontal speed returns to the command speed of $20ft/s$.

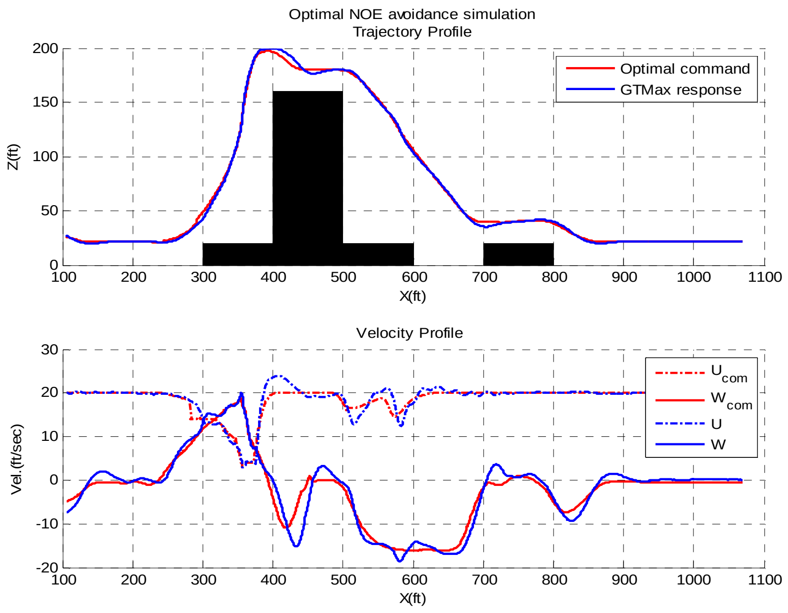


Figure 34: Optimal vertical avoidance example: $V_{cmd} = 20ft/s$ and sensor saturation.

3.3.3 Flight Test

The flight tests of the receding horizon vertical obstacle avoidance were conducted three times. In the first flight test, the trial for avoidance flight using LIDAR was aborted because of the false measurement from the sensor. LIDAR gave unidentifiable noisy outputs near the vehicle. Later it was presumed that the pollen density in the air of the spring season affected the highly sensitive sensor giving very noisy measurements.

The second test was attempted in fairly clear weather, and the sensor did not react to the ambient air conditions. At first, an open loop test was done to check the sensor measurement and real-time trajectory generation, then the closed-loop test followed. The test results are presented in Figures 35 through 38. At the second trial of the test, after avoiding the primary target, a tree of $60ft$ in height, the vehicle suddenly began to sink to the ground, failing to follow the command trajectory, and the pilot had to manually take over the vehicle, barely preventing a hard crash. Figure 37 shows that the vehicle could not follow the command trajectory after the avoidance of the tree. In addition to the incident of the test, the sensor could not measure the actual frontal shape of the tree at the nominal measurement range of the sensor while the vehicle was approaching the tree. Actually it detected the height of the middle of the tree at a distance of about $100ft$, which was half of the nominal range of the sensor. This decreased range and the inability to detect the frontal region of the tree in the far distance resulted in delaying the avoidance less than to $100ft$ before the obstacle. When the vehicle finally reached the tree, the sensor could measure the actual height of the front of the tree, resulting in the shift of the command height by the command modification logic in the framework. These shifts of command height can be seen in both avoidance trials in Figures 35 and 37. The reason for this latency and inaccuracy of laser measurement was not fully investigated, but similar inaccurate measurements, especially for trees without dense leaves, have occurred repeatedly in follow-up flight tests.

The third flight test of vertical obstacle avoidance was conducted at the Mckenna MOUT site in Fort Benning, Georgia, in late June 2011. The avoidance took place three times over a group of small trees about $25ft$ in height with a clearance of $50ft$. The horizontal speed was chosen as $10ft/s$. The test results are presented in Figures 39 and 40. As seen in Figure 39, the command height adjustment over the tree occurred because of sensor measurement, similar to the previous flight test. Overall, the vehicle followed the command trajectories successfully, resulting in the terrain following over the contour line of trees.

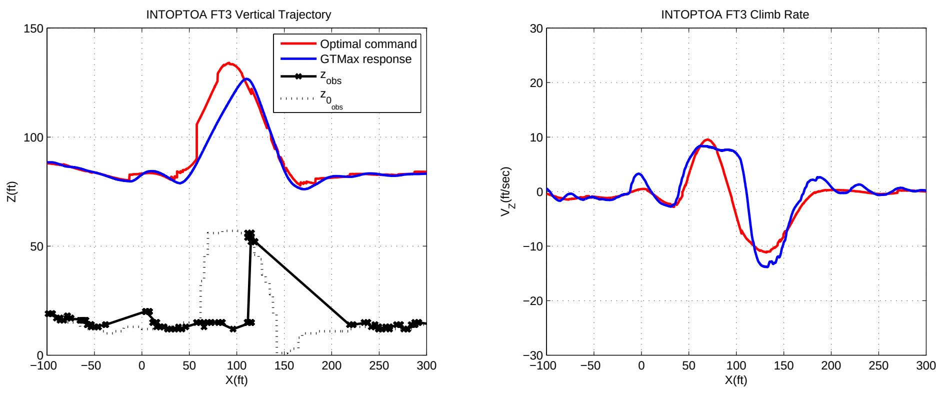


Figure 35: Flight test results of vertical trajectory (left) and climb rate (right): sensor measurement (black mark) could not detect the frontal contour of tree (dot).

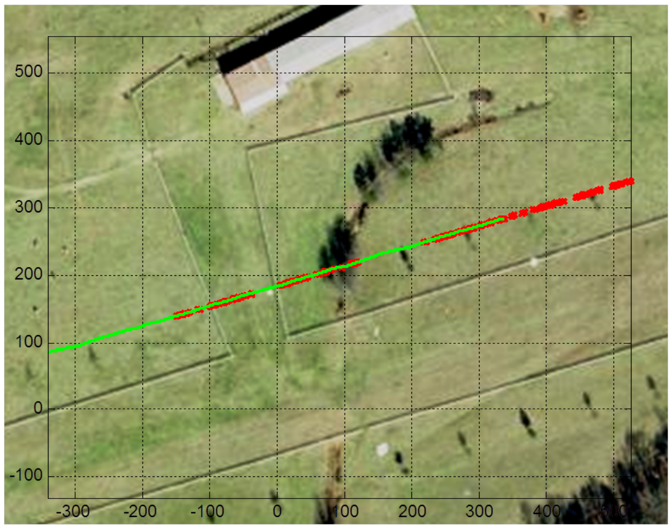


Figure 36: Horizontal trajectory and sensor hit (red marks) of the first trial

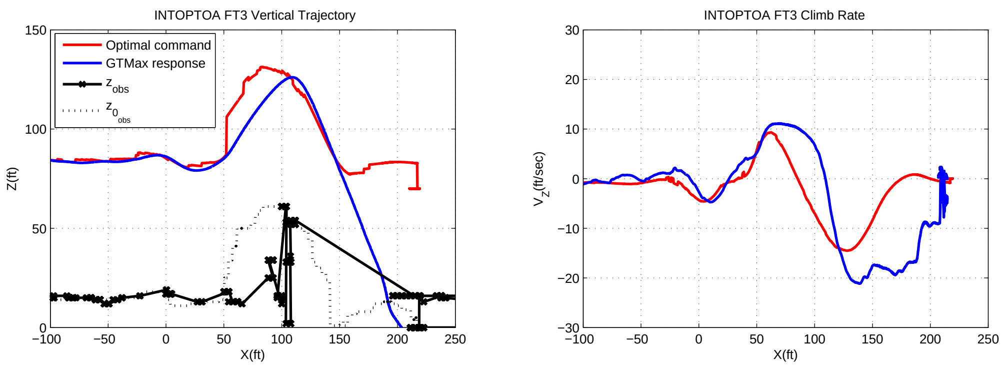


Figure 37: Last trial of flight test: sensor measurement was similar to the first trial; during rapid descent after the tree, the vehicle suddenly sank with a higher rate of descent than the command and actually touched the ground.

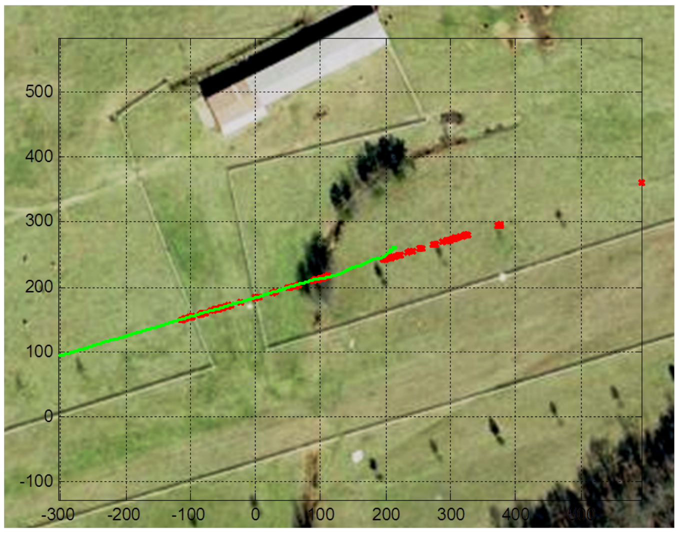


Figure 38: Horizontal trajectory and sensor hit (red marks) of the last trial

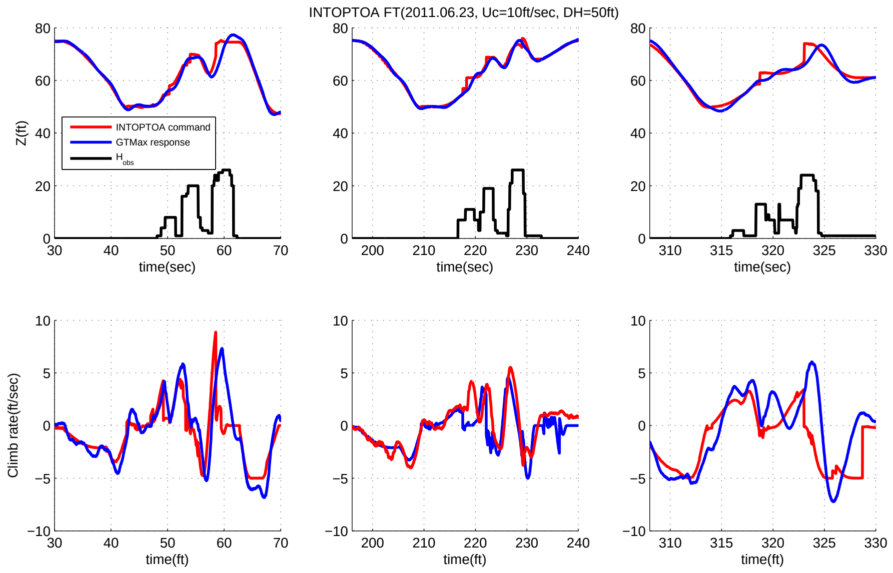


Figure 39: Flight test at McKenna MOUT site: vertical response trajectories (upper) and climb rate responses (lower); command trajectory jumps occurred over the tree line

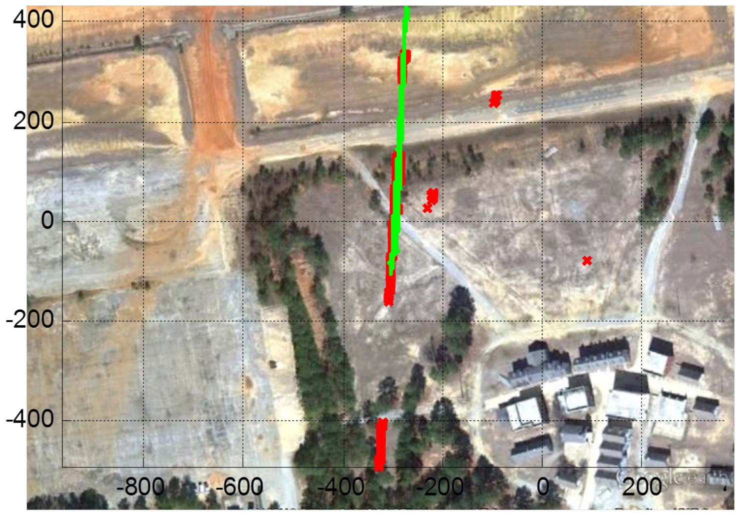


Figure 40: Flight test at McKenna MOUT site: horizontal path and sensor hits.

3.4 Summary

The preliminary studies on optimal obstacle avoidance and their results were presented in this chapter. Using trajectory optimization for optimal obstacle avoidance is the main idea of this thesis. The basic approach evolved from the past study on a single time-optimal avoidance for a virtually provided rectangular obstacle. The time-optimal avoidance of a single obstacle, multiple rectangular obstacles arranged in line along the path, and the arbitrary terminal manifold were studied. Stemming from the past approach, the first practical implementation of INTOPTOA, the integrated optimal obstacle avoidance framework, for vertical trajectory optimization with the actual sensor integration was developed. INTOPTOA is a receding horizon trajectory-planning framework for obstacle avoidance, and it is also applicable to terrain following over unknown environments. The basic approach of INTOPTOA, problem formulation, and the flight test results are presented in this chapter. Unlike the simulation environment, the framework encountered problems associated with the obstacle measurements in actual flight, which have to be taken into account in future practical implementation of the framework: false measurement due to the ambient air conditions, inaccurate measurements of trees at a distance depending on tree shape and density, and sudden changes of measurement near to the obstacle.

CHAPTER IV

LIMIT DETECTION FOR CLIMB RATE

4.1 Estimation of Maximum Climb Rate

Theoretically the maximum rate of climb can be estimated from the excess engine power which is the power difference between available and required power. The required power of a helicopter can be roughly estimated by the general approximation method presented in Appendix A. For the Yamaha RMAX, the base airframe of the Georgia Tech UAV test-bed, this study uses the required power obtained from analytic simulations presented in [105], as shown in Figure 41. It is known that the maximum

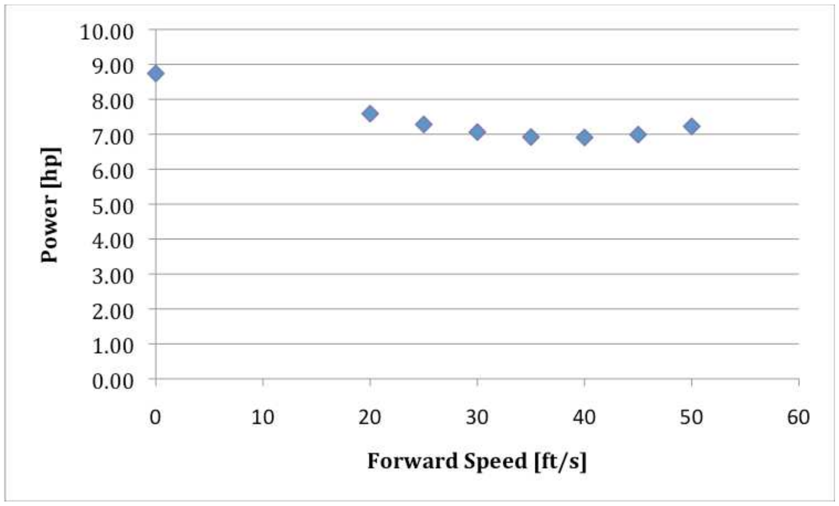


Figure 41: Estimated required power of Yamaha RMAX in forward flight

uninstalled power of the RMAX engine is $21hp$. Assuming 15 percent loss of the installed power, the maximum power output recorded in the simulation model was $18hp$. With this power output and a vehicle empty weight of $166lb$, the maximum vertical speed would be $40ft/s$. If the vehicle weight is increased to $200lb$ to account for extra payload, the corresponding maximum rate of climb decreases to $30ft/s$, and for a $250lb$ vehicle, it decreases to $25ft/s$.

The climb rate limit was determined to be $30ft/s$ for the first flight test by this approximation. However, as shown in Figures 21 through 27 in the previous chapter, the flight test results implied that the actual climb rate limit was lower than the estimated value. It was assessed that the decreased climb rate was caused by the excess power being lower than the estimated value. The excess power can be affected by many factors including: weight, drag, deterioration of engine power, air temperature, etc. In order to account for these factors, a limit detection method was developed to change the limit of the climb rate during flight.

4.2 Effects of Overestimated Climb Rate Limit

With the climb rate limit being used as a constraint within the trajectory optimization, w_U in Equation (76), an inappropriate value of this limit can cause degradation of safety during climb. If an excessive limit value is used, the trajectory optimizer might generate an untraceable command trajectory, especially when the avoidance involves a large obstacle. In addition, the overestimated limit value usually delays the beginning of climbing so the situation may become worse, decreasing the safety of the maneuver.

Figure 42 is a simulation example showing how a dangerous situation could develop when an excessive limit is used. In the simulation, the obstacle height was $180ft$ and the clearance was set to $20ft$. Maximum available engine power was $11,500lbf \cdot ft/s$ but it was arbitrarily decreased to $6,300lbf \cdot ft/s$ to induce a significant degradation in climb performance. With $11,500lbf \cdot ft/s$ power, a climb rate limit of $30ft/s$ was attainable, and hence this value was used in the trajectory optimization for generating the command trajectory. As seen in the simulation result, the actual climb rate was saturated at $18ft/s$ due to the restricted power of $6,300lbf \cdot ft/s$. This lowered the actual climb path to less than the clearance height during the climb maneuver, resulting in the vehicle touching the obstacle.

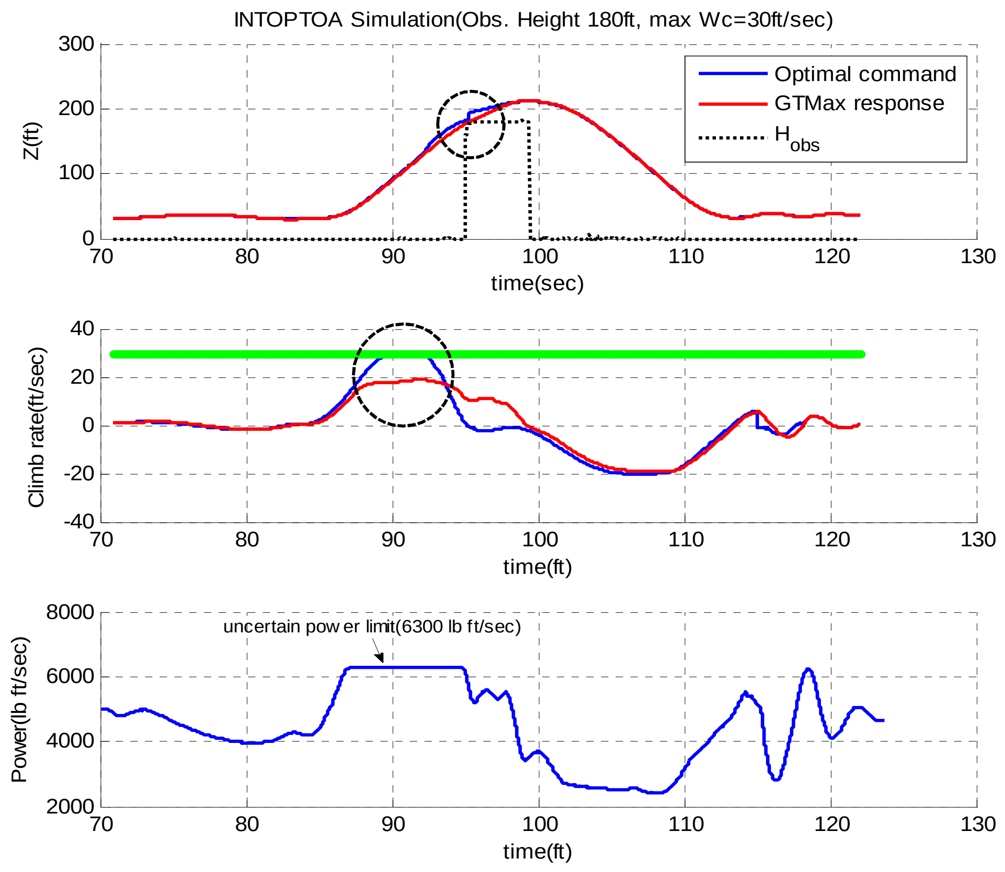


Figure 42: Simulation result with uncertain decrease of engine power: the climb rate limit is set to $30ft/s$ (green line) while the power reaches the arbitrarily chosen maximum power $6,300lbf \cdot ft/s$; the vehicle's climb rate is saturated at $18ft/s$, resulting in collision with the virtual obstacle.

A similar situation happened during the actual flight test as given by Figure 24. As seen in the simulation result, the use of an overestimated climb rate limit in the trajectory optimization might induce untraceable commands. Therefore, detecting the moment when the climb rate becomes saturated and adjusting the current limit value used in the trajectory optimization accordingly are useful in preventing an undesirable command generation, especially in vertical avoidance of large obstacles.

4.3 Detection of Climb Rate Limit

4.3.1 The Adaptive Controller of the Vehicle

Figure 43 shows the overall block diagram of the adaptive controller. The controller is a typical model reference adaptive controller composed of the outer-loop and the inner-loop, and each loop consists of a reference model, a proportional-derivative (PD) compensator, approximate dynamic inversion, a pseudo-control hedging (PCH) logic, and an adaptive neural network (ANN). The PD compensator is augmented to the output of the reference model for the vehicle to track the reference model. The PD compensator for the vertical motion loop is given by:

$$a_{pd} = R_d(\dot{z}_r - z) + R_p(z_r - z) \tag{83}$$

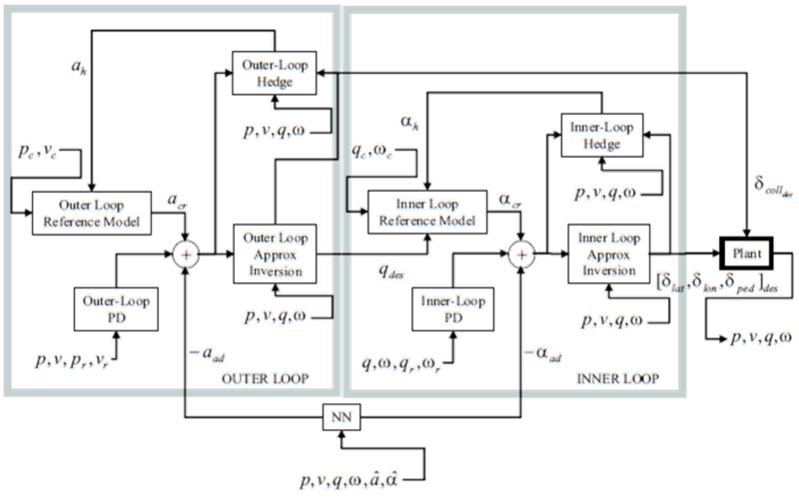


Figure 43: Block diagram of the adaptive controller [62]

The adaptive controller is designed to make the vehicle follow the dynamic response of the reference model while compensating for uncertainties in the model inversion using the ANN. In the event of actuator saturation, the pseudo control hedging (PCH) shown as outer-loop hedge or inner-loop hedge is used to remove any excess command that has caused the actuator saturation. When the vehicle is not in any limit saturation, such as actuator or performance limit, the vehicle motion is expected to be similar to the reference model, and thus, the PD compensator signal a_{pd} should asymptotically go to zero. Otherwise, if the given commands are not perfectly traceable, the PCH block of the controller internally adjusts the reference model output to be followed by the vehicle. The PCH reduces the reference model output when the actuator command is saturated, and thus, it prevents the ANN from adapting to input nonlinearities arising out of actuator saturation. The ANN generates the adaptation signal needed to compensate for the tracking error arising due to the use of an approximate model for model inversion.

The working of the adaptive controller can be seen in Figure 45 for the case of normal vertical climbing. Figure 46 is for the case of vertical climbing with reduced engine power. As seen in Figures 45 and 46, the adaptive controller of the vehicle internally modifies the external commands to traceable reference commands, so the error in tracking of the reference model by the vehicle is maintained within a small boundary (see the subplots for tracking errors in Figures 45 and 46).

4.3.2 Limit Detection Logic

Figure 44 depicts the overall structure of the proposed climb rate limit detection logic. Vertical position and rate commands are shaped through a second-order model and the shaped commands are subtracted from the corresponding vehicle outputs to obtain the position and climb rate errors. The model used for the commands shaping in the limit detector is similar to that of the adaptive controller except that the hedging part of the adaptive controller is not included in the limit detector. The vertical position and climb rate errors are combined to obtain a composite error, $\hat{a}_{pd}$, as shown in Figure 44. The composite error is passed through a first-order filter (with a time constant of $\tau = 0.1sec$ for this study), and the output of the filter, $\bar{a}_{pd}$, is used for the detection of the event when the vehicle climb rate is at its limit. When the vehicle climb rate is at its limit, the vertical position and climb rate errors in the limit detection logic increase well above their nominal values, resulting in a sharp increase in the filtered composite error $\bar{a}_{pd}$. Once this error goes above a preset threshold during a climb phase of the vehicle trajectory, the detection logic triggers to decrease the current climb rate limit value at a preselected rate, and the modified lower value of the climb rate limit is used in the trajectory optimization.

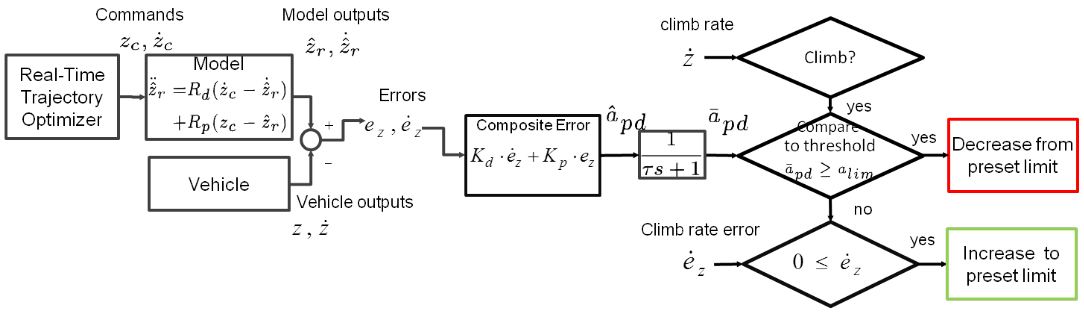


Figure 44: Climb rate limit detection logic

4.3.3 Simulation Results

As described in the previous sub-section, once the limit detection is triggered during climb, the current limit value is reduced linearly at a preselected rate to the current climb rate, whereas, if the detector is not triggered during an aggressive climb, the current limit value is gradually increased, again at a preselected rate to its nominal value. Figure 47 shows the effect of the logic wherein the lowered climb rate limit by the detection logic enables the safe avoidance of the obstacle for which the avoidance failed without the climb rate limit modification as seen previously in Figure 42.

Figure 48 presents another simulation result in which the available power is artificially lowered in order to simulate the case of the vehicle reaching the climb rate limit during obstacle avoidance maneuvers. As shown in the fourth sub-plot of Figure 48, the power available is lowered from an initial value of $11,500 lbf \cdot ft/s$ to $8,500 lbf \cdot ft/s$, thus reducing the available excess power for the climb. As the climb rate limit value used in the trajectory optimizer corresponds to the available power of $11,500 lbf \cdot ft/s$, the reduction in power results in climb rate saturation, causing the filtered composite error, $\bar{a}_{pd}$, (shown as the dotted red line in the third sub-plot of Figure 48) to exceed the preset threshold of $15ft/s^2$. This triggers the limit detection logic to lower the climb rate limit value (shown as the green line in the second sub-plot of Figure 48). The reduced value of the climb rate limit is subsequently used in the trajectory optimization, resulting in a safe avoidance of the first obstacle shown in Figure 48. After passing the first obstacle, the filtered composite error stays well below the threshold of $15ft/s^2$, thus triggering a slight increase in the climb rate limit during the avoidance of the second obstacle. Prior to the avoidance of the third obstacle, the power available is further reduced to $6,500lbf \cdot ft/s$. Since the current value of the climb rate limit in the trajectory optimization is no longer valid with the reduced available power, the filtered composite error signal, $\bar{a}_{pd}$, once again goes above the preset threshold of $15ft/s^2$, causing the limit detection logic to decrease the climb rate limit to the saturated climb rate of the vehicle. For comparison, the a_{pd} signal from the vertical loop of the adaptive controller is also shown in the third sub-plot of Figure 48. As expected, the magnitude of the a_{pd} signal from the adaptive controller stays low as it represents the tracking error between the hedged reference model and the vehicle.

4.4 Summary

Accurate climb rate limit, which is closely related to the excess power of rotary-wing aircraft, is important in the trajectory generation for low-altitude flights such as terrain following and nap of the earth. A conservative lower limit value of the climb rate limit produces vertical flight profiles that may be safe but may also result in less aggressive obstacle avoidance maneuvers. On the other hand, the use of a higher estimate of the climb rate limit can increase the aggressiveness of the command trajectory but may result in the vehicle being unable to follow such a command trajectory, thus, creating an unsafe situation for obstacle avoidance.

A general method for power estimation, which is summarized in Appendix A, may be applied to estimate the climb rate limit, but it also requires accurate knowledge on the related parameters such as weight, drag, and induced velocity, etc. In order to determine the saturation of climb rate and adjust the climb rate limit value for its use in trajectory optimization during obstacle field navigation, a simple limit detection logic is proposed and is evaluated in simulation using the GUST.

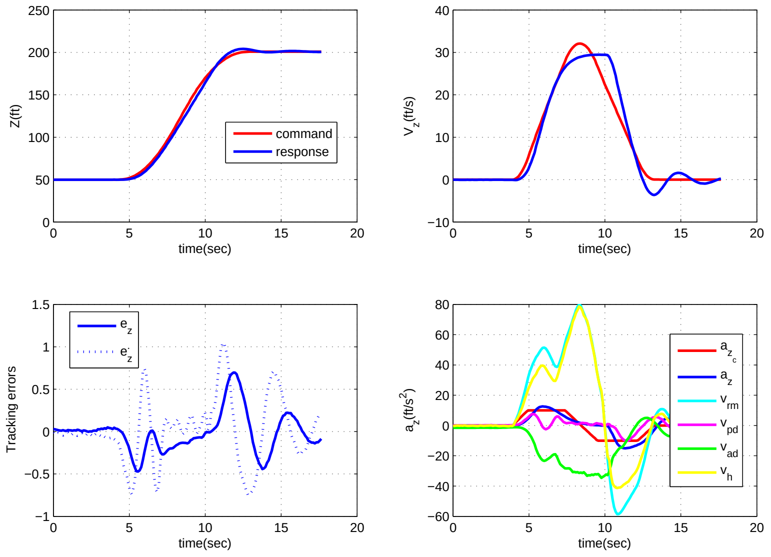


Figure 45: The control of vertical climb in normal conditions: the adaptive controller follows the reference model; height command from $50ft$ to $200ft$ with max. acceleration of $10ft/s^2$ and speed of $50ft/s$; ν_{rm} is the reference model output, ν_{pd} is the PD compensator output, ν_{ad} is the ANN output, and ν_h is the PCH output.

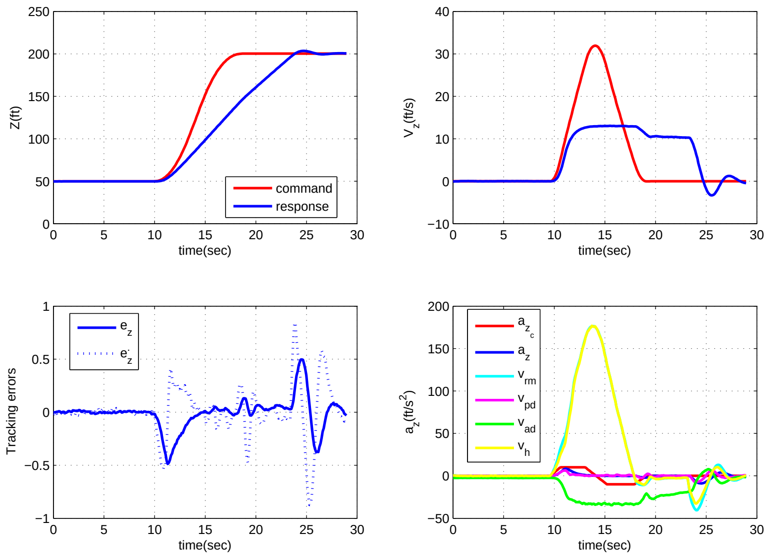


Figure 46: The control of vertical climb in abnormal conditions: the available power is reduced to 60 percent of the nominal; the adaptive controller hedges the reference model not to follow the command (top row) generating the followable reference model output; errors (bottom-left) between the reference model and the feedback.

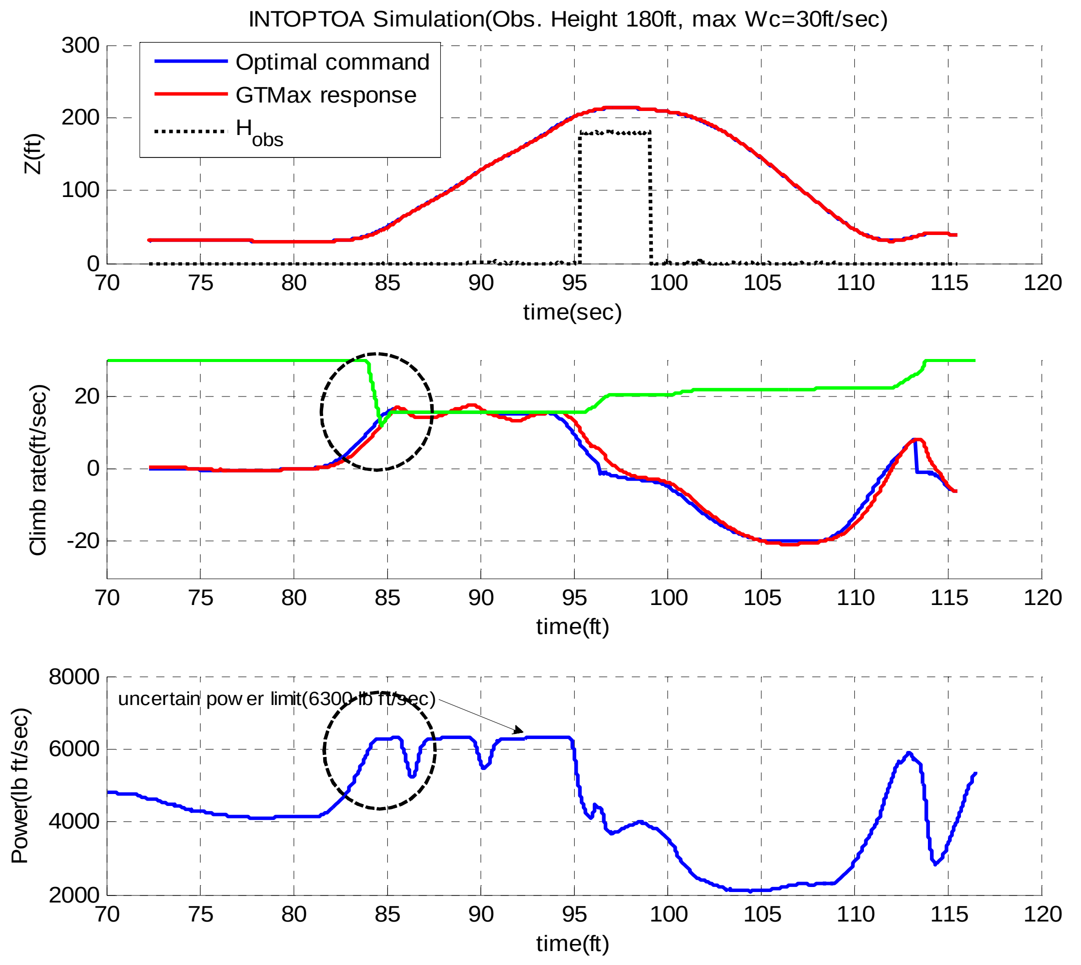


Figure 47: Simulation with climb rate limit detection logic: the climb rate limit is originally set to $30ft/s$ (green line); when the vehicle's climb rate begins to saturate ($18ft/s$) the limit value is lowered for safe avoidance of the same obstacle of Figure 42.

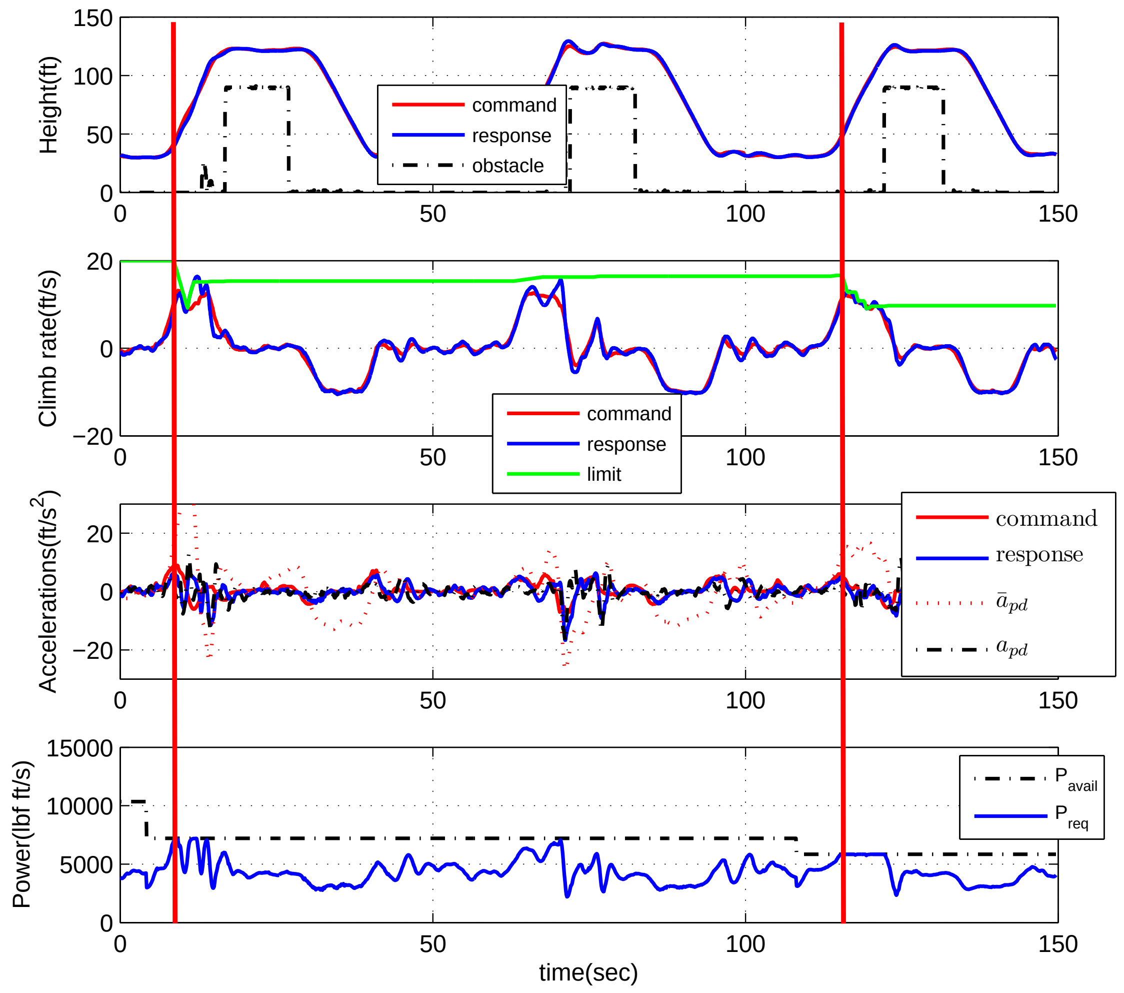


Figure 48: Simulation with engine power variation: the max. available power is lowered to $8,500 lbf \cdot lb/s$ and then $6,500 lbf \cdot lb/s$ during simulation; when the required power reaches the max. power during climb, the detection logic triggers the decrease of the climb rate limit.

CHAPTER V

INTEGRATED FRAMEWORK FOR OBSTACLE AVOIDANCE

5.1 *Three-Dimensional Avoidance*

The trajectory optimization for obstacle avoidance was extended to three-dimensional (3D) avoidance within a finite sensor range as depicted in Figure 49. The scope

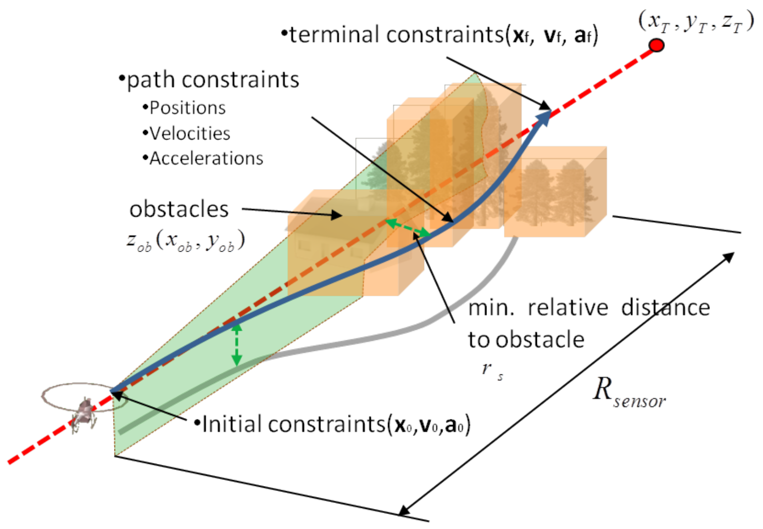

•terminal constraints($\mathbf{x}_f$, $\mathbf{v}_f$, $\mathbf{a}_f$)
(x_T, y_T, z_T)
•path constraints
•Positions
•Velocities
•Accelerations
obstacles
$z_{ob}(x_{ob}, y_{ob})$
min. relative distance to obstacle
r_s
R_{sensor}
•Initial constraints($\mathbf{x}_0$,$\mathbf{v}_0$,$\mathbf{a}_0$)

Figure 49: Scope of receding horizon trajectory optimization

of the trajectory optimization is to find the local optimal command trajectory that minimizes the position deviation from the preplanned straight path to the destination while maintaining the clearance from the terrain and obstacles for safety, as well as remaining feasible within the maneuverability limits and the vehicle dynamics. The problem is a typical receding horizon (RH) trajectory optimization problem of which the objective is to find an open-loop optimal control within the sensor range. The optimization proceeds with the current states at every sample time and the resultant optimal control is continuously updated until the vehicle reaches the target position.

RH trajectory optimization has been widely applied to the trajectory planning for UAVs, as the optimization can nicely describe the trajectory planning problem. However it presumes that the optimization problem is solvable in real time to control or guide the vehicle. In theory, the infinite horizon optimal solution is amenable in the case of searching for the best path for the mission objective; however, in practice for the problems concerning complex and uncertain obstacle environments, the infinite horizon optimization can quickly become computationally intractable for real-time computation. In addition, it becomes unnecessary without full knowledge of the obstacle field beyond the sensor range.

RH trajectory optimization has obvious advantages: it is definitely computationally less demanding than the infinite horizon optimization, the problem is solvable in real time, and the solution satisfies subjected constraints as well as vehicle dynamics. RH optimization solves the local optimums of finite horizon at each sampling time, which is computationally feasible to onboard computation in real time and has little impact on closed-loop stability. However, it should be noted that the sequence of local optima do not guarantee the global optimum over the full horizon. So, it has been a key issue in RH optimization, especially for trajectory planning purposes, on how to take into account the discarded tail of the horizon in the problem formulation. To resolve this issue, past studies on MPC or RHC have used a cost-to-go (CTG) function [90] or a special final value function such as Control Lyapunov Function (CLF) [60] as an extra terminal cost in the optimization problem.

An interesting fact is that many studies on the optimization-based trajectory planning considered the combination of a global path searching and a local trajectory planning in their approaches to resolve the finite horizon issue. A CTG map can be obtained from off-line computation over the roughly described entire field of obstacles by using simple path searching methods or computationally demanding indirect optimization methods, and it is used in the online local trajectory computation to ensure finding the global optimal path. Kuwata [78] used a similar concept to get the 3D optimal path over the cuboid obstacles using a CTG map obtained in offline base computation over a known field of obstacles. The use of CTG or other value functions have been considered beneficial to MPC-based trajectory planning, provided that the remaining portion of the entire horizon is at least roughly known.

Unlike the above approaches, this study assumes that there is no obstacle beyond the sensor range while the vehicle flies through the unexplored obstacle field unless the obstacle cuboids database is provided a priori. From this base assumption, the optimization problem is formulated to minimize the position deviation from the straight path to the target position.

5.2 Problem Formulation

The 3D trajectory optimization for obstacle avoidance considered the integrated weighted quadratic distance to the target position to be minimized as given by

$$J = \int_{t_0}^{t_f} \left[\left(\frac{(x_T - x(t))}{R_x} \right)^2 + \left(\frac{(y_T - y(t))}{R_y} \right)^2 + \left(\frac{(z_T - z(t))}{R_z} \right)^2 \right] dt \tag{84}$$

where (x_T, y_T, z_T) is the target position and R_x, R_y, R_z are constants to penalize the position deviation along each axis. By adjusting these constants, the cost function can induce different avoidance patterns: a relatively smaller value of R_x compared to R_y and R_z makes the trajectory similar to the time-optimal pattern in which the optimization minimizes the longitudinal distance from the target primarily, whereas the opposite setting of the penalty constants can minimize the lateral and vertical deviation from the straight line to the target, usually resulting in increased proximity to the obstacle surfaces. So, it is necessary to determine the penalty constants relevant to meeting the mission objectives and the permissible agility of the vehicle.

By default, the next waypoint-to-go is selected as the target position. So, if the vehicle is initially located sufficiently far from the waypoint, the cost function drives the vehicle to minimize the distance initially accelerating to the commanded horizontal speed and keeping the flight direction aligned with the initial line of sight to the targeted point. As the vehicle approaches the waypoint, a waypoint switching logic is used to switch the target waypoint with the subsequent waypoint of the preplanned mission data. Another feature related to the target position is that it might be selected arbitrarily and could even be the moving position of a different vehicle. So using the proposed cost function with the appropriate set of constraints can transcribe the trajectory optimization problem to the problem of position tracking or the formation flight. Position, velocity, and acceleration are defined in the guidance frame, which is an inertial frame of which the x-axis is aligned in the direction of the next waypoint from the previous waypoint and its origin is located on that of the inertial frame as shown in Figure 10. The trajectory is subjected to the kinematics and simplified 1st order dynamics as given by Equations (69) through (71).

For obstacle avoidance, the trajectory is subjected to several constraints. Obviously, the clearance from the obstacle geometry $\mathcal{O}(x_{ob}, y_{ob}, z_{ob})$ is the primary path constraint; the position of the vehicle should be outside the boundary of the obstacle with clearance distance at any instance expressed as:

$$r_{clr}(\vec{p}(t)) \triangleq \min_{\vec{p}_{ob} \in \mathcal{O}} \|\vec{p}(t) - \vec{p}_{ob}\| \geq R_s \tag{85}$$

where $\vec{p}_{ob} = [x_{ob}, y_{ob}, z_{ob}]^T$ is the obstacle surface coordinate, $\vec{p}(t)$ is the current vehicle position, $\|\cdot\|$ denotes the Euclidean norm, and R_s is the safety clearance that can be determined arbitrarily by the user considering the trade-off between safety and allowable agility during avoidance.

As noted earlier, the vehicle performance and envelope limits need to be met in order to protect the structural components of the vehicle from damage. Some of those limits can be transcribed to the limits on the motion variables such as velocities and accelerations and can be formulated as path constraints on velocity and acceleration. This thesis considered constraints on velocity components, the total speed, and the total acceleration limit as

$$
\begin{gathered}
0 \leq u \leq u_U \\
v_L \leq v \leq v_U \\
w_L \leq w \leq w_U \\
V_{t_L} \leq \sqrt{u^2 + v^2 + w^2} \leq V_{t_U} \\
0 \leq \left(\frac{a_x}{a_{xmax}}\right)^2 + \left(\frac{a_y}{a_{ymax}}\right)^2 + \left(\frac{a_z}{a_{zmax}}\right)^2 \leq 1
\end{gathered}
\tag{86}
$$

Here, the vertical speed limits w_L and w_U need to be selected carefully with consideration for the power limit and the aerodynamic stability in descent at a low-horizontal speed, especially for rotary-wing UAVs. A low limit of the total speed constraint is needed to prevent an unnecessary decrease of flight speed during an avoidance maneuver.

The forward flight speed at the terminal point is left free. The terminal longitudinal position is determined by the measured obstacle horizon which is basically the farthest detected obstacle range plus margin or the maximum sensor range in the case that no obstacle is measured. The lateral and the vertical terminal position is left free but it should be outside the measured obstacle geometry having greater clearance than the safety clearance. The terminal constraints are summarized as

$$
\begin{gathered}
t_f = free \\
x_f = \begin{cases} x_{ob_{detected}} \\ x_0 + R_{sensor} \end{cases}, y_f = z_f = free \\
0 \leq u_f \leq u_U \\
v_L \leq v_f \leq v_U \\
w_f = 0 \\
\vec{a}(t_f) = free \ \text{ or } [0\ 0\ 0]^T
\end{gathered}
\tag{87}
$$

where R_{sensor} is the finite sensor range. Terminal acceleration can be set free or can be zero for the level and non-accelerating flight condition at the end of the finite horizon. In general, during mid-course trajectory optimization or for the avoidance of large obstacles, free terminal acceleration might be beneficial for finding a more aggressive trajectory. On the other hand, enforcing zero acceleration at the finite horizon can make the trajectory leveled at the end and it may initiate an avoidance maneuver at an earlier point from an obstacle. Simulations with both terminal conditions of acceleration were carried out for the benchmark cases discussed later in this chapter.

5.3 Integrated Optimal Obstacle Avoidance Framework

Figure 7 depicts the overall structure of the framework, named as INTOPTOA, which is composed of the RH trajectory generator and the vehicle controller. The RH trajectory generator is connected in parallel with the existing base planner, and if the sensor detects obstacles the RH trajectory generator overrides the base planner. Figure 50 below shows the detailed view of the framework structure in avoidance mode.

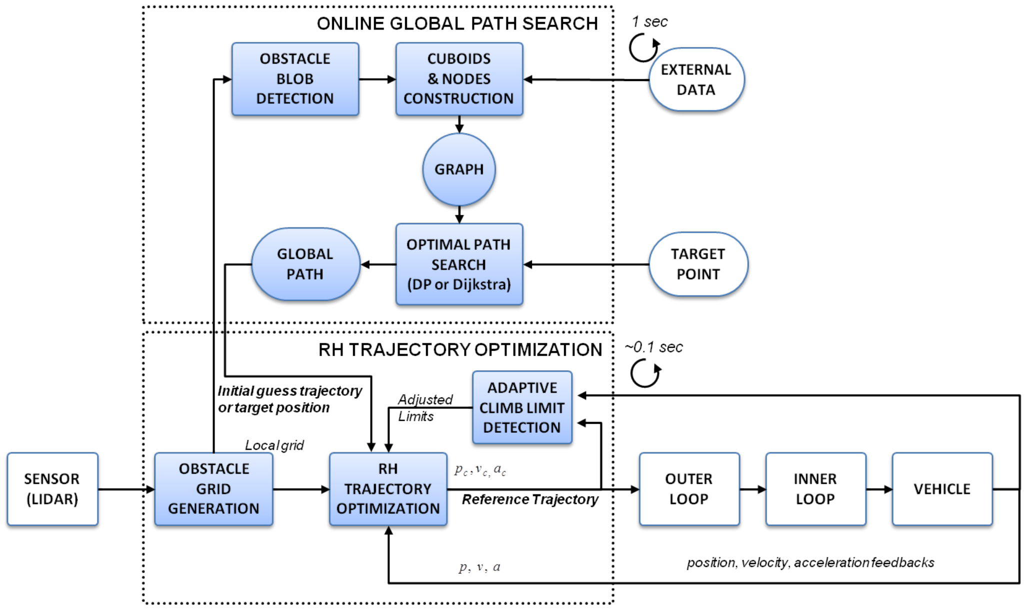

ONLINE GLOBAL PATH SEARCH
1 sec
OBSTACLE BLOB DETECTION
CUBOIDS & NODES CONSTRUCTION
EXTERNAL DATA
GRAPH
GLOBAL PATH
OPTIMAL PATH SEARCH (DP or Dijkstra)
TARGET POINT
RH TRAJECTORY OPTIMIZATION
~0.1 sec
Initial guess trajectory or target position
Adjusted Limits
ADAPTIVE CLIMB LIMIT DETECTION
Local grid
SENSOR (LIDAR)
OBSTACLE GRID GENERATION
RH TRAJECTORY OPTIMIZATION
p_c, v_c, a_c
Reference Trajectory
OUTER LOOP
INNER LOOP
VEHICLE
p, v, a
position, velocity, acceleration feedbacks

Figure 50: INTegrated OPTimal Obstacle Avoidance (INTOPTOA) structure

When the vehicle is in obstacle avoidance mode, the RH trajectory generator continuously computes and updates the command trajectory by solving the optimization problem at the current state, and the command trajectory is followed by the adaptive controller, shown as the outer-loop and the inner-loop in Figure 50. As already noted, the two-layer architecture has the advantage that the trajectory generator does not need to consider the accurate dynamics of the vehicle, uncertainties, and measurement errors because the adaptive controller maintains the stability and follows the command trajectory as closely as possible while compensating for sensor measurement noise, unmodeled dynamics, and other disturbances. However, it is true that only suboptimal solutions are attainable, and there might be some loss in performance, but the loss in optimality and performance might be compensated for by not having an undesirable impact on the closed-loop system stability.

As shown in Figure 50, the trajectory generator is actually a hybrid trajectory planner that combines the RH trajectory generation module with the global path searching module. The combination of the global and the local trajectory planning is nothing new; it has already been attempted in other research in different combinations. However, the fundamentals of the approaches are almost the same:*to complement the incompleteness of the local trajectory planning and the dynamical infeasibilities of the global path planning.*

Other approaches in the past MPC-related studies [6, 27, 78, 90] mostly used the combination of offline global path searching in a known obstacle field to obtain the offline cost-to-go values or the rough global path for the template, and then they used the MPC path planner to find the local optimal path using the cost-to-go values or to adjust the global path to be dynamically tractable. The hybrid method of this thesis is different: the global path is computed online using the optimal graph searching over the approximated obstacle field, represented as cuboids, by the use of dynamic programming, and the resultant coarse global path is used as the initial guess for the RH trajectory optimizer. The coarse path can be used otherwise to determine the intermediate target point for the local trajectory optimization. Details of the method are described in the following sections.

5.4 Obstacle Grid Generation

Unlike the past framework in Chapter 3 which only considered the vertical profile of the obstacle field in front of the vehicle, this framework includes the obstacle grid generation module that constructs the surface of the local area in front of the vehicle using the measured obstacle data for 3D avoidance trajectory generation. The use of a sensor is an essential requirement for the framework to provide the obstacle avoidance capability in unknown flight environments, thus a 2D scanning LIDAR was selected as the sensor. LIDAR produces a point cloud of returned laser-hit positions inside a scanned region.

5.4.1 Grid Mapping

The obstacle grid generation is basically the mapping of the point cloud on the local grid, the area of which is defined by 100 × 100 units of distance moving with the vehicle as illustrated in Figure 51. The coordinate frame of the grid is a moving frame, the x-axis of which is aligned to the guidance frame. The unit distance is set by the user considering the maximum sensor range.

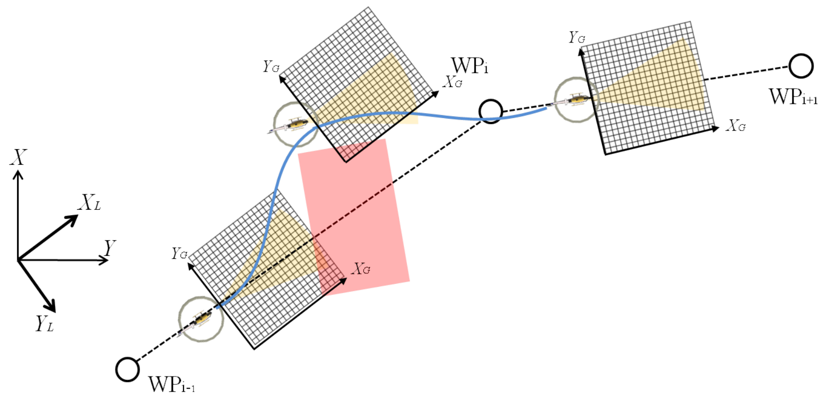


Figure 51: Concept of obstacle grid construction

The position vectors in the point cloud array from the sensor are represented as the vectors with respect to the inertial frame, thus, the grid mapping is done by the coordinate transformation from the inertial to the grid coordinates. Every data point in the grid is shifted in the opposite direction of the vehicle velocity as it moves a grid unit distance. Once a point is shifted out of boundary it is completely erased.

The LIDAR is mounted on the front of the vehicle to scan vertically as shown in Figure 8. Thus, in order to measure 3D obstacle geometry, the vehicle is commanded to make an oscillatory yawing motion with a given frequency and yawing angle. The sampled point cloud is mapped on the grid with a safety boundary and is continuously accumulated on the grid if a measured point is inside the grid. The purpose of the safety boundary and filling the actual laser hit points inside the boundary are to increase the safety of avoidance, making the avoidance on the roughly represented grid surface wrap the actual obstacle with a margin. Figure 52 illustrates this concept of obstacle grid mapping and filling. Every point of the point cloud is mapped and accumulated on the grid with a discretized horizontal position, so only the highest point at the same grid position is saved on the grid array. This also contributes to the conservative construction of the obstacle grid.

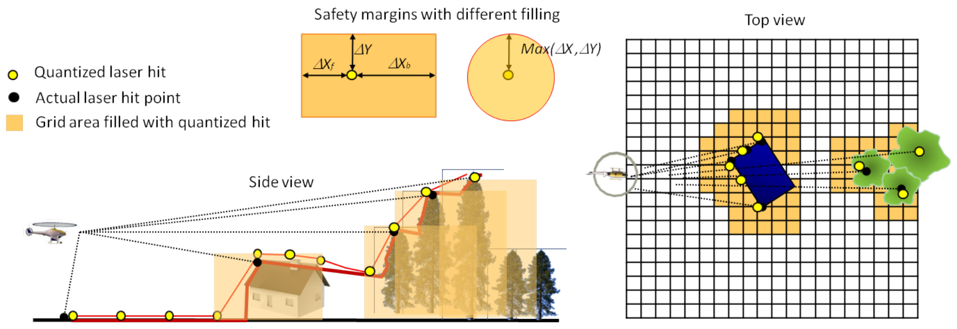


Figure 52: Obstacle grid mapping with safety margin

Once the obstacle grid is filled, every grid point is averaged with the neighboring points within a specified local window to smooth the grid surface.

$$z_{ob}^{f}(x_g, y_g) = \frac{1}{4ab} \sum_{i=-a}^{a} \sum_{j=-b}^{b} z_{ob}(x_g + i, y + j) \tag{88}$$

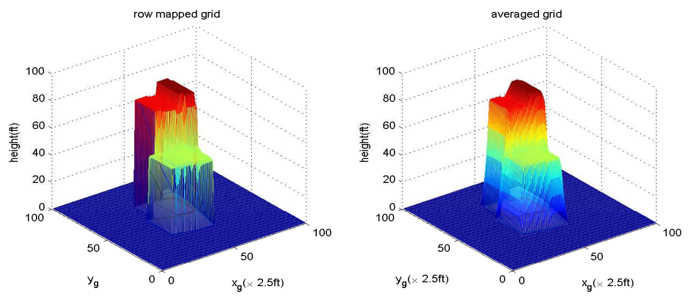
where a, b are the integer values that determine the averaging window by $(a \times b)\delta_r$ and δr is the grid unit distance. The obstacle grid otherwise can be filtered with a simple 1st order spatial filter:

$$\begin{aligned} z_{ob}^f(x_g+1, y_g) &= \alpha z_{ob}(x_g+1, y_g) + (1-\alpha) z_{ob}^f(x_g, y_g) \\ z_{ob}^f(x_g, y_g+1) &= \alpha z_{ob}(x_g, y_g+1) + (1-\alpha) z_{ob}^f(x_g, y_g) \\ \alpha &= \frac{\delta r}{\delta r + \tau_r} \end{aligned} \tag{89}$$

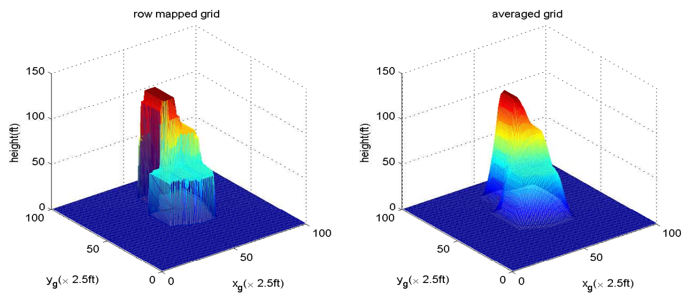
where τ_r is the spatial distance of filtering greater than zero.

Either filtering or averaging is beneficial to the trajectory optimization, which requires the gradient of the obstacle surface for the numerical procedure. A smoother gradient of the obstacle surface can be beneficial to the convergence of the numerical optimization procedure. Figure 53 is an example of a raw obstacle grid map and the averaged map from the raw grid during a simulation, and Figure 54 is for the case of 1st order filtering of the raw grid. Figures 53 and 54 show that averaging can produce a smoother obstacle surface but it may overly blunt the corners of the original grid map. Thus, careful determination for averaging parameters are required so as not to lose significant obstacle features, considering the expected characteristics of obstacles in the mission field.

Obstacle grid generation considered in this thesis is highly dependent on the robust measurement of the sensor, so it actually may be vulnerable to false measurements. Although the spatial filter and the averaging can smooth out a spiky false measure, unlike other occupancy map generation methods based on the probabilistic approaches [3, 27], the confidence of the sensor measurement and its time variation cannot be accounted for in the current approach. However, for conservative avoidance that places more significance on safety than agility and pursues a low proximity to the obstacle requiring accurate and robust measurements, this approach might be sufficient.

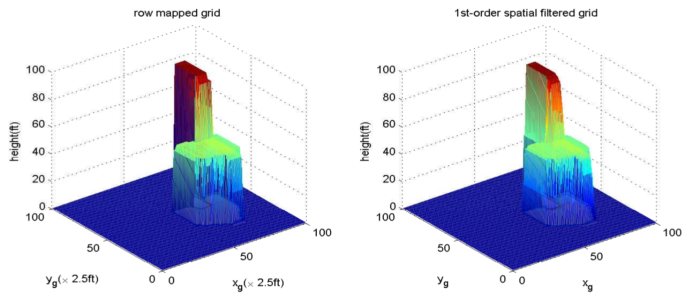


(a) averaging window ($15ft \times 15ft$), $\delta r = 2.5ft$

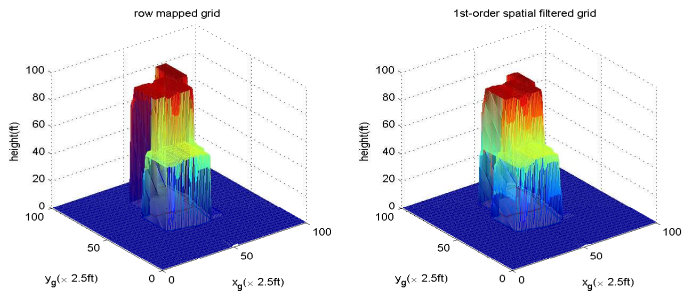


(b) averaging window ($30ft \times 30ft$), $\delta r = 2.5ft$

Figure 53: Raw grid (left) and averaged grid (right)



(a) $\alpha = 0.143$, $\delta r = 2.5ft$



(b) $\alpha = 0.077$, $\delta r = 2.5ft$

Figure 54: Raw grid (left) and 1st-order spatial filtered grid (right)

5.4.2 Cuboid Obstacle Construction

The smoothed or filtered obstacle grid is mainly used for the numerical optimization by the RH trajectory generation module. Additionally, the obstacle grid is used to construct simplified obstacle cuboids for the global path searching. Figure 55 illustrates the procedure of the online cuboid obstacle construction and Table 2 summarizes the details of the procedure.

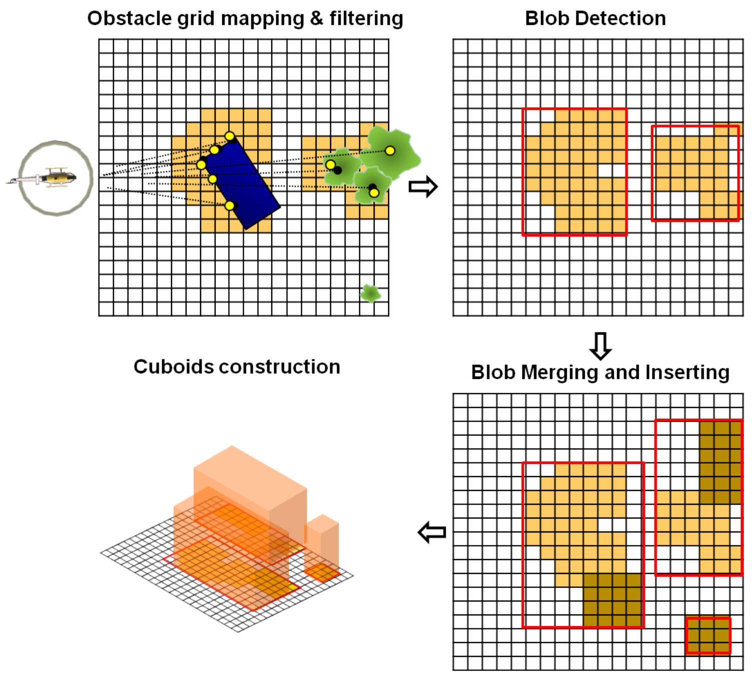


Figure 55: Online procedure of cuboid obstacle construction

At every sampling time of the global path search module, 1*sec* in this study, the current filtered obstacle grid is processed by the obstacle blob detection module to extract the rectangular regions of obstacles on the grid. The blob detection by thresholding is used for this process. Once a rectangular obstacle region is extracted, the height and the area of the region are processed to construct the cuboid, and it is compared to the cuboids recorded in the database. Depending on the areas and the locations, the newly constructed cuboid is to be either inserted into the database or merged with one of the existing cuboids, and the database module in the frame manages the change of existing database or the creation of new items. Using a cuboid to represent an obstacle is mainly to facilitate the global path searching with a lower computational load: the database can be easily constructed and managed, the data structure is simple, the processing is easy, and the amount of memory for the large area of the obstacle field can be reduced so the transfer of the whole database can be fast. However, the obstacle geometry can be over-simplified, and it might be inappropriate for representing obstacles like long slanted wires or walls.

Table 2: Online obstacle cuboid creation procedure

```
Algorithm: Blob detection and cuboid database construction

1. Convert the current grid map into the image array of the height of obstacles.
2. Perform blob detection and create the detected blob lists.
3. Retrieve a blob sequentially from the list and check the area overlapping with obstacle cuboids saved in the database.
    (a) if the blob is inside the existing cuboids, check the height and merge to the first cuboid.
    (b) if the blob is outside or less than the specified overlap ratio to all existing cuboids, insert into the database as a new cuboid.
    (c) if the blob is greater than the specified overlap ratio to one of existing cuboids, merge it with the cuboid.
4. Save changed database.
5. Clear and destroy the current blob lists and the image array
```

5.5 Global Path Generation by Graph Searching

In Graph theory, a graph is a mathematical entity defined as a pair of vertices connected by an edge, and it has been commonly used for network problems like the path searching of this thesis. In this thesis, *directed graph searching* is used to find the global path to the destination while detouring all obstacles, represented as cuboids.

The obstacle cuboids constructed from the obstacle grid are saved as a database and the database is continuously updated during flight unless it is reset by the operator. Once any obstacle cuboid is created online or transferred from external sources, the global path searching goes forward with the procedures: the node construction around cuboids, validation of feasible graphs, and optimal path searching through the valid graphs. Figure 56 is an illustration that briefly explains the procedure, and the detailed algorithm is described in Table 3.

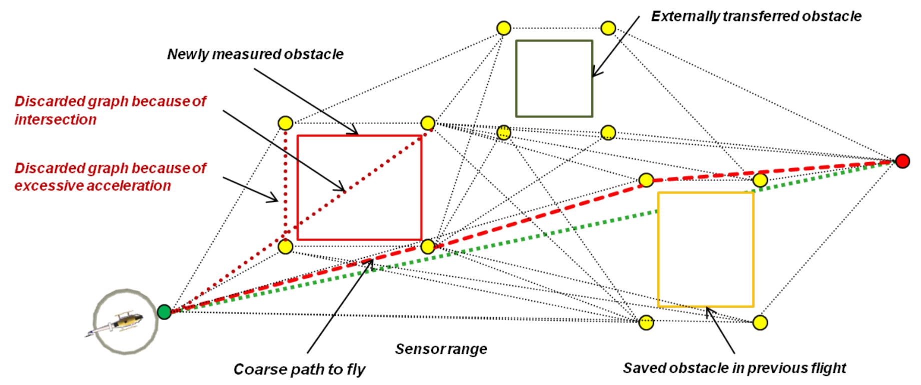
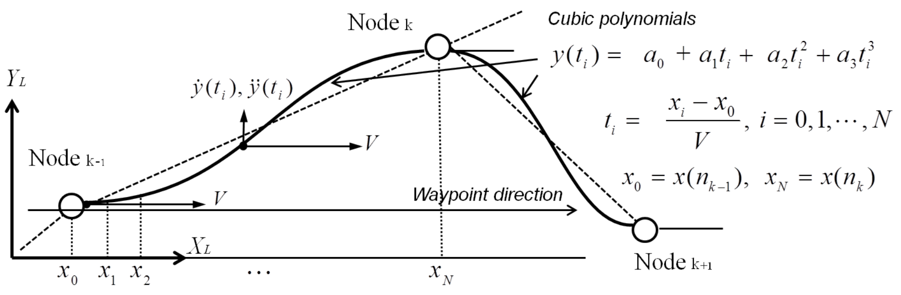

Externally transferred obstacle
Newly measured obstacle
Discarded graph because of intersection
Discarded graph because of excessive acceleration
Sensor range
Coarse path to fly
Saved obstacle in previous flight

Figure 56: Brief illustration of graph searching

5.5.1 Feasibility Check Using Cubic Polynomial Approximation

The feasibility of any graph is verified in two ways, by checking the intersection by line and rectangle and by approximating the unit path by cubic polynomial to check the acceleration limit. It is obvious that the global path determined by the graph search is represented by the lines connecting nodes from the start (current vehicle position) to the destination. Because the dynamic feasibility cannot be checked in the graph search, the resultant path can easily become partially dynamically infeasible between the nodes if its length is short and the angle between the line connecting the nodes and the line to the destination is large. The unit path approximation using C^2 cubic polynomial between two nodes enables a rough check on the feasibility of any unit graph.



Node k
Cubic polynomials
$y(t_i) = a_0 + a_1 t_i + a_2 t_i^2 + a_3 t_i^3$
$\dot{y}(t_i), \ddot{y}(t_i)$
Y_L
$t_i = \frac{x_i - x_0}{V}, \ i = 0,1,\cdots,N$
V
Node k-1
$x_0 = x(n_{k-1}), \ x_N = x(n_k)$
V
Waypoint direction
X_L
Node k+1
x_0 x_1 x_2 $\cdots$ x_N

Figure 57: Cubic polynomial approximation of graphs

With the assumption that the approaching speed to the next node is constant, $V = const.$, and the entry and the exit angles are the same as the angle between the line connecting nodes and the line of waypoints, as shown in Figure 57, the y trajectory between the nodes can be represented as a cubic polynomial

$$y(t_i) = a_0 + a_1 t_i + a_2 t_i^2 + a_3 t_i^3 \tag{90}$$

where $(x_0, y_0) = (x(n_{k-1}), y(n_{k-1}))$, $(x_N, y_N) = (x(n_k), y(n_k))$, $t_i = \frac{x_i - x_0}{V}$, $i = 0, 1, \cdots, N$, and a_0, a_1, a_2, a_3 are determined by the boundary conditions.

$$\begin{aligned}
a_0 &= y_0 \\
a_1 &= y_0' \\
a_2 &= \frac{3(y_N - y_0) - (2y_0' + y_N')t_f}{t_f^2} \\
a_3 &= \frac{2(y_N - y_0) - (y_0' + y_N')t_f}{t_f^3} \\
t_f &= \frac{x_N - x_0}{V}
\end{aligned} \tag{91}$$

So, the lateral acceleration can be approximated as a simple linear equation by differentiating twice

$$\ddot{y}(t_i) = 2a_2 + 6a_3 t_i,\ i = 0, \cdots, N \tag{92}$$

and the maximum acceleration occurs at both ends. Assuming that the initial and terminal directions are aligned to the waypoint direction, $y'_0 = y'_N = 0$, the maximum lateral acceleration is simply given as

$$\max|\ddot{y}(t_i)| = 2a_2 = 2\frac{3(y_N - y_0)}{t_f^2} = \frac{6(y(n_k) - y(n_{k-1})V^2}{(x(n_k) - x(n_{k-1}))^2} \tag{93}$$

The vertical acceleration also can be checked similarly by simply changing y positions of the nodes to z positions. Therefore, the possible peak acceleration of any graph can be easily checked. However, it is a rough estimation based on the assumption of constant speed and the same angles at both ends. Therefore, it is appropriate to give some margins in the feasible acceleration of the unit graph so as not to overly invalidate graphs.

5.5.2 Dynamic Programming

The optimal graph searching is done by dynamic programming (DP), which has been widely used in the search for the global path in probabilistic roadmap methods and in visibility graph methods. The theory originated from Bellman's *principle of optimality*, which simply states that *if a trajectory is optimal, the end-portion of it also should be optimal* [7]. So, in order to use DP to find the optimal path, each graph should be evaluated for cost values. DP actually works as a backtracking recursive optimal cost value finder and node selector based on the imposed cost values on graphs, as mathematically expressed as:

$$f(n_k) = \min_{n_k \in S(n_k)} [c_{k,l} + f(n_l)] \tag{94}$$

where $f(n_l)$ is the cost value at a previous optimal segment of the path, so by backtracking it is set to zero initially, $c_{k,l}$ is the cost of a graph of node k and l, $S(n_k)$ is the set of selectable nodes including n_k, and $f(n_k)$ is the next optimal cost at optimal node n_k. In the case of finding the shortest path, $c_{k,l}$ becomes the length of a graph or time to travel. By checking the intersection and the violation of the acceleration limit as given in the previous section, the infeasible graph is set to have a large cost value so as not to be selected.

Table 3: Global path searching procedure

Algorithm: Graph construction and global path searching

```
1. Create nodes around and at the top of the cuboids with clearance distance.
2. Sort the nodes with respect to the distance from current position.
3. Find the nodes pair or group of which each node is close to others within
   a specified distance and leave one node and eliminate the rest.
4. Adjust the node height if it is inside other cuboids.
5. Construct the graph array based on the final node set.
6. Compute the cost for each feasible pairs of nodes, and during computation
   check the feasibility and assign the cost.
    (a) Check the node pairs that intersect any obstacle cuboid, then assign
        a big number for its cost value in the graph array.
    (b) Check the node pairs that may exceed the acceleration limit, then
        assign a big number, based on the cubic polynomial approximation
        of the path between nodes.
7. Run dynamic programming to find the optimal sequence of nodes.
8. Construct a path array by interpolating the selected node positions.
9. Clear the nodes and the graph array.
```

5.6 *Software Implementation*

INTOPTOA was integrated into the Georgia Tech UAV Simulation Tool (GUST) and the onboard software of the rotary-wing UAV test-bed shown in Figures 58 and 59.

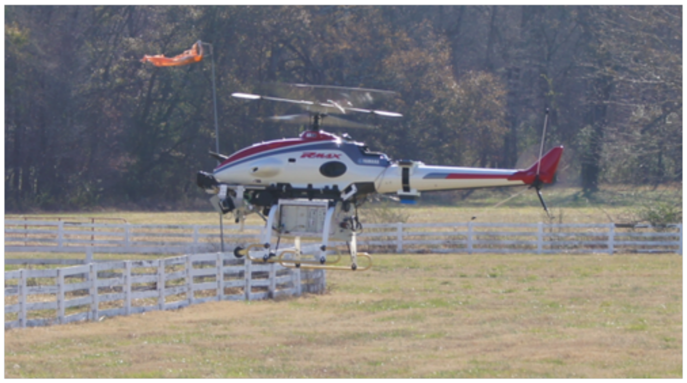
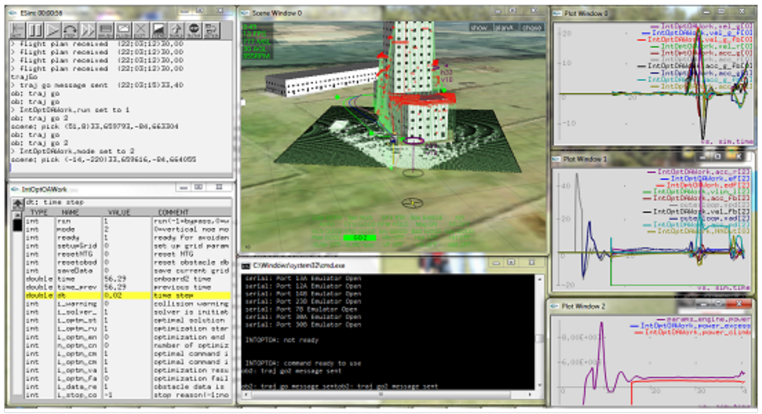


Figure 58: UAV test-bed at Georgia Tech: The base airframe is Yamaha RMAX.

Software implementation in GUST uses C/C++ programming. GUST can provide a real-time simulation environment, hardware-in-the-loop, and research flight test operations for any software component or hardware that is to be incorporated into the UAVs currently being operated by the Georgia Tech UAV research group.



```
> flight plan received  (22:03:12)30.00
> flight plan received  (22:03:12)30.00
> flight plan received  (22:03:12)30.00
> flight plan received  (22:03:12)30.00
trajGo
> traj go message sent  (22:03:15)33.40
ob: traj go
ob: traj go
> IntOptOAWork.run set to 1
ob: traj go 2
scene: pick (51,0)33.659793,-84.663304
ob: traj go
ob: traj go 2
> IntOptOAWork.mode set to 2
scene: pick (-14,-220)33.659616,-84.664055
```

```
dt: time step
TYPE    NAME         VALUE   COMMENT
int     run          1       run(-1=bypass,0=w
int     mode         2       0=vertical nos mo
int     ready        1       ready for avoidan
int     setupGrid    0       set up grid param
int     resetMTG     0       reset MTG
int     resetobsd    0       reset obstacle db
int     saveData     0       save current grid
double  time         56.29   onboard2 time
double  time_prev    56.29   previous time
double  dt           0.02    time step
int     i_warning    0       collision warning
int     i_solver_    1       solver is initiat
int     i_optm_st    1       optimal solution
int     i_optm_ru    1       optimization star
int     i_optn_en    0       optimization end
int     n_optn_cn    0       number of optimiz
int     i_optn_cm    1       optimal command i
int     i_optn_cm    1       optimal command i
int     i_optn_va    1       optimization resu
int     i_optn_fa    0       optimization fail
int     i_data_re    1       obstacle data is
int     i_stop_co    -1      stop reason(-1:no
```

```
serial: Port 10A Emulator Open
serial: Port 12A Emulator Open
serial: Port 14B Emulator Open
serial: Port 23B Emulator Open
serial: Port 7B Emulator Open
serial: Port 30A Emulator Open
serial: Port 30B Emulator Open

INTOPTOA: not ready


INTOPTOA: command ready to use
ob2: traj go2 message sent

ob2: traj go message sentob2: traj go2 message sent
```

Figure 59: A captured screen during a simulation in GUST

GUST includes models of the sensors, aircraft, and aircraft interfaces, down to the level of binary serial data, i.e., packets, so it enables injection of model error and environmental disturbances directly into the aimed software component or hardware components. It includes a flexible scene generation capability and re-configurable data communication routines, enabling a large number of possible hardware-in-the-loop simulation configurations [63].

The UAV is outfitted with two onboard computers, named Onboard1 and Onboard2. Onboard1 is for vehicle guidance and control, and Onboard2 is for in-flight computations of experimental functionalities such as visual camera image processing, vision-based target tracking, formation flight, etc. In order to isolate the vehicle stability and control from the risk of computation delay or unexpected computational loop crash during the optimization, INTOPTOA is implemented in Onboard2. Figure 60 shows the current onboard system integration.

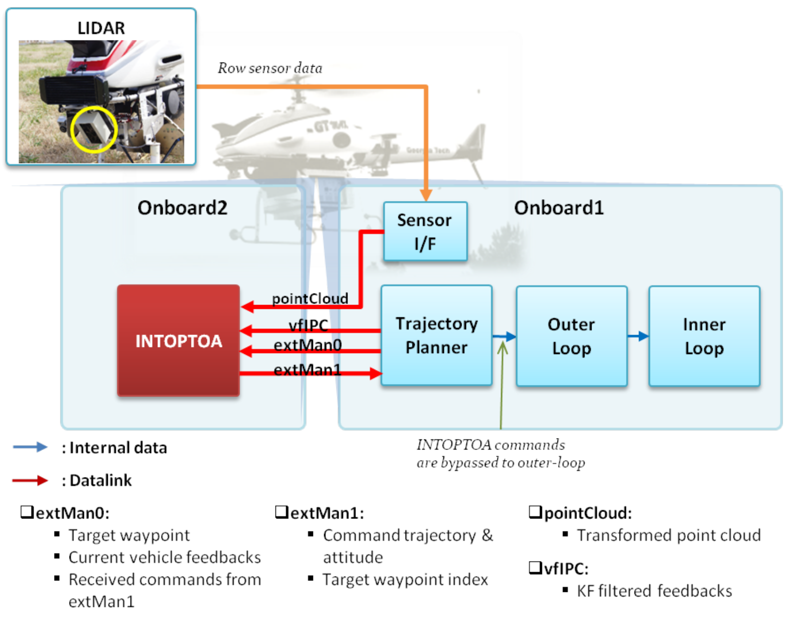


Figure 60: The onboard system integration of INTOPTOA

INTOPTOA receives vehicle navigation data, that is, position, velocity, acceleration, attitude, and attitude rate, via data-link. The scanned raw point cloud data from the sensor, LIDAR (Sick LD-MRS), is filtered and sampled by the sensor interface module in Onboard1 and transmitted through a separately assigned data-link channel. The command trajectory computed by INTOPTOA is transmitted to Onboard1 through a data-link and can bypass the base planner in Onboard1.

An important feature of the onboard implementation of INTOPTOA is that the RH trajectory optimization module of Figure 50 is programmed as a multi-threaded routine, in order to run the time-demanding optimization process in separate threads of the Onboard2 processor. This is an important feature for the RH trajectory optimization to work for its original purposes: *to use previous computed commands until the commands are updated*, as well as to maintain safe control of the vehicle. The difference between the single-threading and multi-threading is briefly illustrated in Figure 61 below.

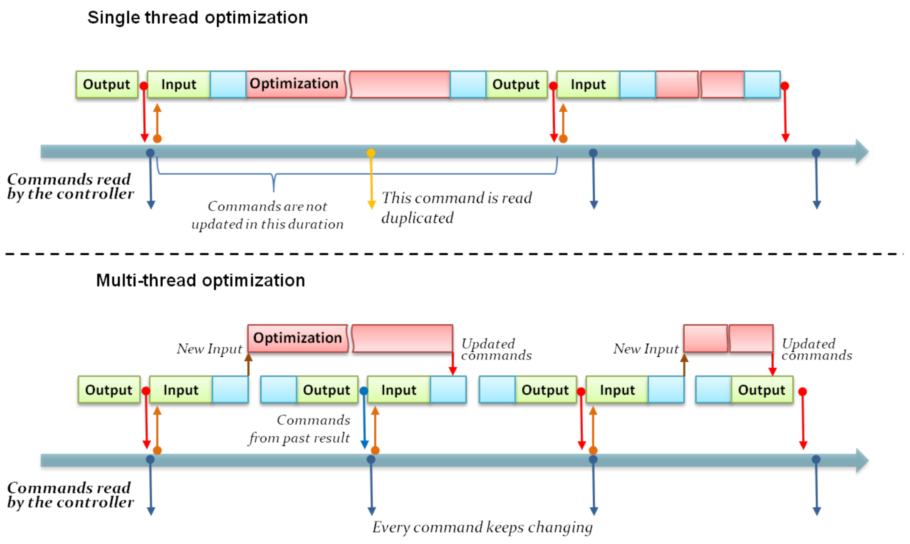

Single thread optimization
Output
Input
Optimization
Output
Input
Commands read by the controller
Commands are not updated in this duration
This command is read duplicated
Multi-thread optimization
New Input
Optimization
Updated commands
New Input
Updated commands
Output
Input
Output
Input
Output
Input
Output
Commands from past result
Commands read by the controller
Every command keeps changing

Figure 61: Conceptual difference in single vs multi-threaded process

As seen in Figure 61, if the RH trajectory optimization module is programmed as a single-threaded routine, it is obvious that the entire Onboard2 process will be held up until the optimization is completed, which means that the command to Onboard1 will be frozen to the last sent until the optimization is finished. This might be acceptable if the computation time is less than a hundred milliseconds. However, even though the real-time optimal solver is used, the computation time for the optimization can often rise up to a second. Depending on the flight conditions such as speed and acceleration, even a few hundred milliseconds of a command freeze may cause a serious problems in command following and eventually impair loop stability. Figure 63 is an example simulation showing how the single thread optimization can result in unsafe fluctuations of vehicle motion.

On the other hand, a multi-threaded optimization can remove this risk completely. In a multi-threaded optimization, the optimization routine is launched with a separate thread, so the Onboard2 processor can handle the computation for optimization and other normal processes simultaneously by the CPU time slicing, which makes the optimization seem to run independently. This is possible if the computer has multiprocessors or the CPU is multi-core. Thus the command can be continuously generated from the previous computation result while the optimization is running and can be updated with new results as soon as the optimization is finished. The effect of multi-threading is obvious when we compare Figures 63 and 64. Multi-threading is not just a means of maintaining loop stability; rather, it is the desired way that any RHC or MPC works. Multi-threading has other advantages that can facilitate computation: threads share memory and devices so unlike multi-processing, with separate memory and devices, it allows rapid sharing of information by the shared memory without inter-process communication protocols, there is no need for manual coding to eliminate pauses due to hardware response, and it can allow messages or signals to be received in the middle of a long computation. However, there is a possibility of serious dead locks, or asynchronous operations. Figure 62 is a brief illustration of the overall process flow between Onboard1 and Onboard2 during the INTOPTOA run. Onboard1 and Onboard2 computers shares data through datalinks. The global path search module in INTOPTOA runs at $1Hz$, the obstacle grid generation module updates the grid at every $0.1sec$, and the optimization module is processed in a separate thread of Onboard2 processor. The the sampling rate varies depending on the computation time for the optimal solution.

Figure 65 is an example 3D avoidance simulation result using INTOPTOA showing the obstacle grid map obtained while the vehicle is passing by. The interim command trajectories are continuously changed during avoidance, and the resultant avoidance trajectory is formed on the concentrated region of the interim command trajectories.

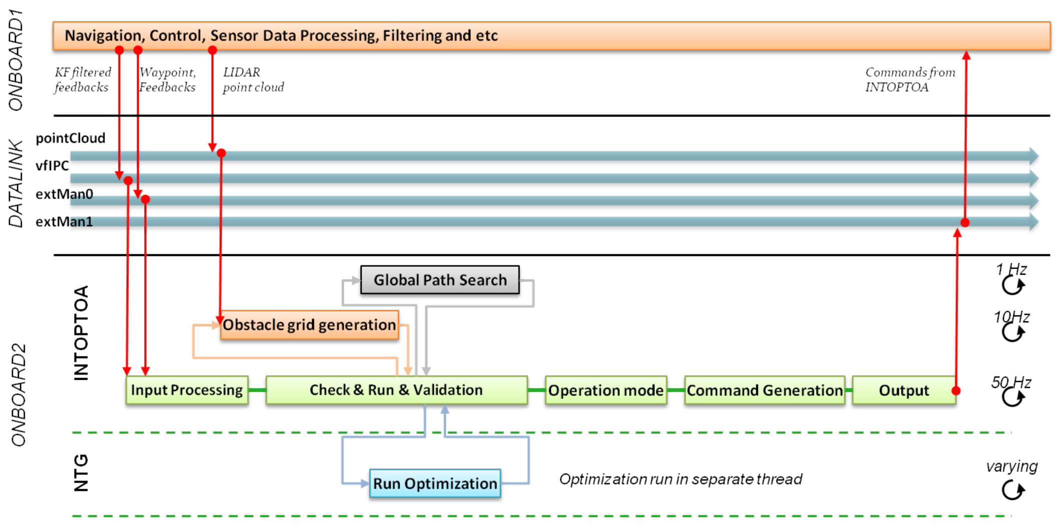


Figure 62: Rough process flow of INTOPTOA

5.7 Summary

The receding horizon optimization was extended to a three-dimensional trajectory planning. First, the method of generating the local receding obstacle grid was introduced to find the local optimal path over it. Then the combined use of global shortest path searching and the receding horizon trajectory optimization was proposed, and for global path searching the local grid map was used to build an approximate obstacle field by cuboid representation. A basic image processing algorithm, blob detection by thresholding, was used to extract the obstacle cuboid from the obstacle grid. The global shortest path was determined by dynamic programming over the cuboid obstacle field and the resultant path was used for the initial guess to the local trajectory optimizer. The final algorithm was implemented into GUST and the onboard software of the UAV test-bed, and to meet the original concept of the receding horizon trajectory optimization, the trajectory optimization module was programmed to run in a multi-threaded routine. The effect of multi-threading was verified in simulation.

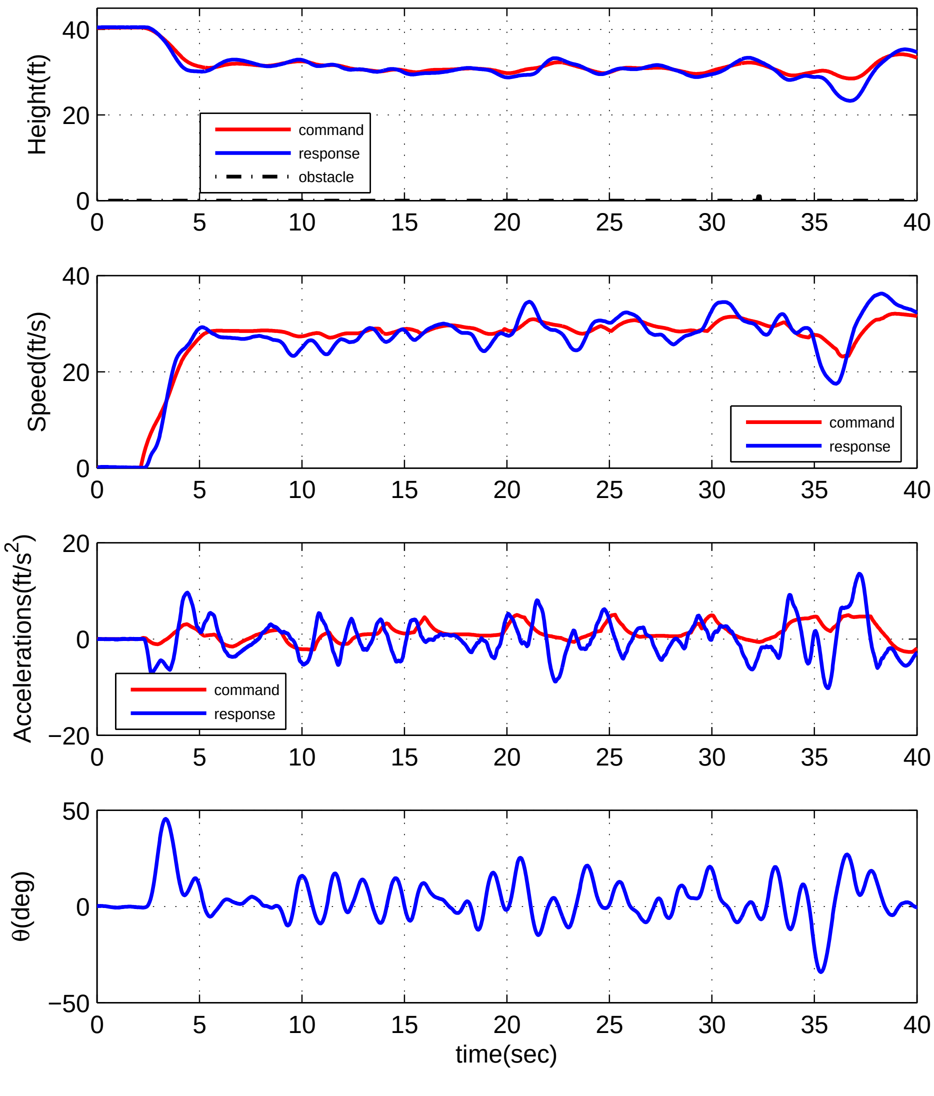


Figure 63: Example level flight simulation with single thread optimization: The overall loop of INTOPTOA and the vehicle controller become unstable.

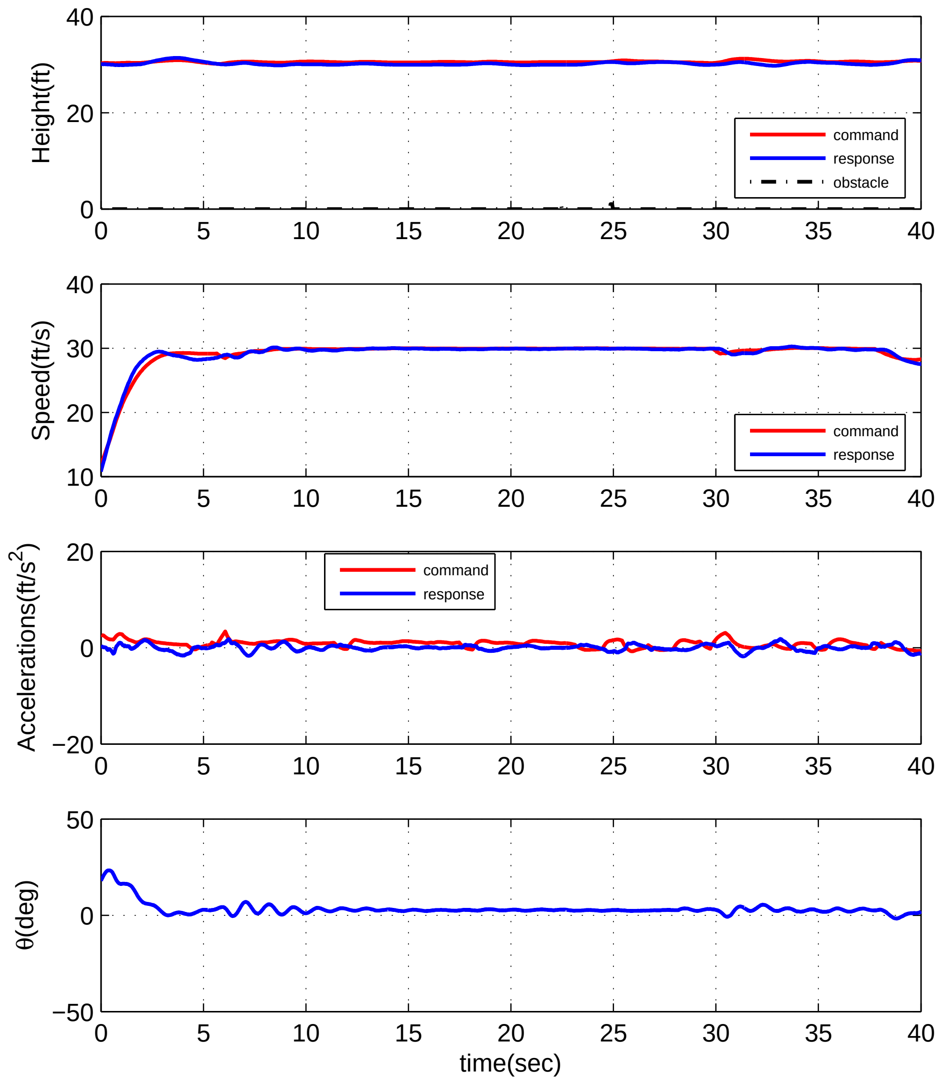


Figure 64: Example level flight simulation with multi-threaded optimization: The overall loop remains stabilized.

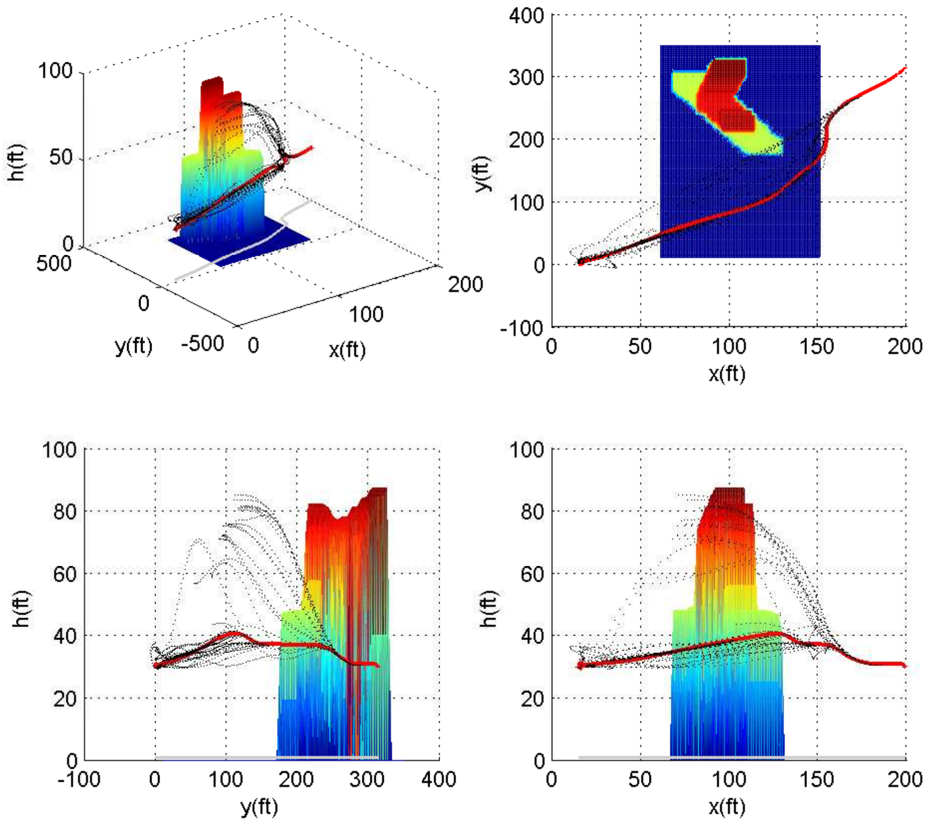


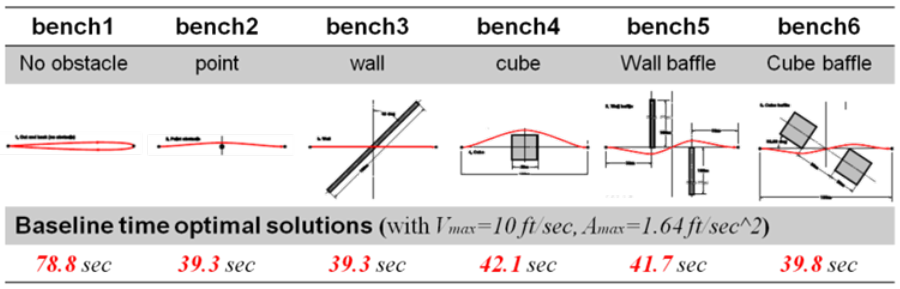
Figure 65: Example 3D obstacle avoidance simulation: Intermediate trajectory command (dotted black lines) are updated as the measured target grid varies.

CHAPTER VI

SIMULATION AND FLIGHT TEST EVALUATION

6.1 Benchmark Tests

In order to test and evaluate INTOPTOA, six simple benchmark tests proposed by Mettler [89] are simulated and four of six benchmark cases are tested in actual flight tests. The summary of the benchmark cases and the baseline time-optimal solutions are given in Figure 66. The benchmark obstacles consisted of geometric obstacle primitives which were designed to exercise and evaluate a given capability of the obstacle avoidance algorithm. The original test configurations described in the reference have an interval distance of $330ft$ from the start point, and all obstacles have a maximum height of $66ft$. Start and end point heights of vehicle trajectory are $33ft$, and the velocity is required to be zero at both start and end points. The vehicle maneuverability limits are set to $v_{max} = 10ft/s$, $v_{vert} = 5ft/s$ and $a_{max} = 1.64ft/s^2$, and the clearance is selected as $25ft$ horizontally and $19.8ft$ vertically.



bench1	bench2	bench3	bench4	bench5	bench6
No obstacle	point	wall	cube	Wall baffle	Cube baffle
Baseline time optimal solutions (with V_{max}=10 ft/sec, A_{max}=1.64 ft/sec^2)					
78.8 sec	39.3 sec	39.3 sec	42.1 sec	41.7 sec	39.8 sec

Figure 66: Benchmark cases and baseline solutions [89]

The benchmark flight tests included only cases 3 through 6. For safety reasons, the base flight altitude was increased to $120ft$ and the virtual sensor measurement simulated in the ground control station (GCS) was transmitted to the onboard computers through the data-link during the flight test. So the actual sensor was bypassed because the test purpose was to evaluate the performance of the optimal trajectory generation and avoidance, not the overall framework including the actual sensor in the real obstacle environment.

6.2 Benchmark Simulations

The performance of INTOPTOA for all six benchmark cases was evaluated by the simulation using GUST and the resultant times from start to arrival at the target point were compared to the baseline solutions from [89]. In fact, the baseline solutions are noted as the accurate time-optimal solutions, whereas the simulation results are obtained by minimizing the integrated weighted quadratic distance to the target point, see Equation (84). Thus, theoretically the baseline solutions should be smaller than the results obtained by INTOPTOA. Another point regarding the simulation results is that although command acceleration and speed are limited to $1.64 ft/s^2$ and $10 ft/s$, respectively, the actual response of the vehicle may not follow the command perfectly, resulting in minor violations of acceleration and velocity constraints. This aspect needs to be considered while evaluating the benchmark test results.

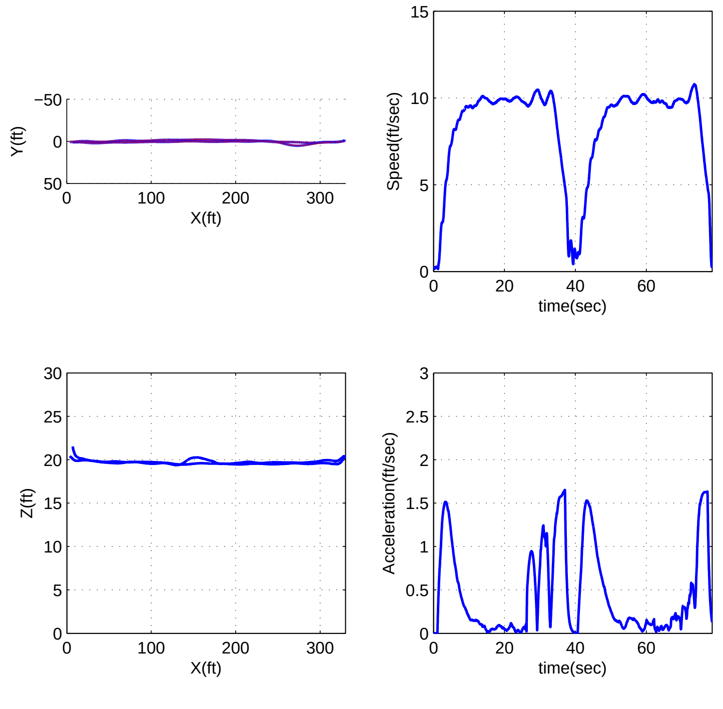
Figures 67 through 72 present the simulation results for the benchmark cases 1 through 6. Trajectory, speed, and acceleration are plotted to show the overall results of each simulation. Table 4 summarizes all the simulation results along with the baseline values of [89]. The times obtained by INTOPTOA are rounded values of the time difference between the start of the maneuver and when the vehicle reached the target position. Simulations are also done with different terminal acceleration conditions, i.e., free and zero terminal accelerations, in order to check the effect of the terminal condition on the convergence of solutions and on the overall avoidance maneuver. It is seen that the use of either of the two terminal acceleration constraints, i.e., either zero or free terminal acceleration constraint, results in converged solutions in real time while the resultant overall trajectories for the two cases are very similar.

Overall, the INTOPTOA produced trajectories similar to the baseline solution even though the optimization did not specifically target a time-optimal solution. However, these results could be attributed to the fact that the obstacle shapes selected were simple. In addition, the performance index (see Equation (84)) does indirectly minimize the time especially when the vehicle is initially positioned on the straight line joining the start and end point.

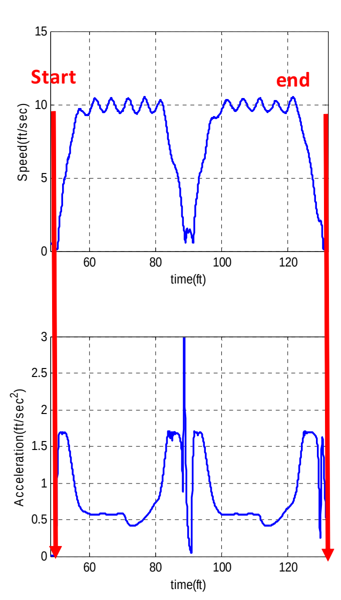
Table 4: Comparison of INTOPTOA to baseline time to maneuver

Cases	Obstacle Type	Baseline [1]	OFN[2]	INTOPTOA (Sim.)[3]
case 1	Out and back	78.8	84.5	80.0
case 2	Point	39.3	49.2	40.0
case 3	Wall	39.3	54.1	40.0
case 4	Cube	42.1	52.2	42.0
case 5	Wall baffle	41.7	52.5	42.0
case 6	Cube baffle	39.8	51.9	40.0

[1] time-optimal solutions by offline computation
[2] Obstacle Field Navigation base on MPA+RHC[89]
[3] simulation results in case of free terminal acceleration

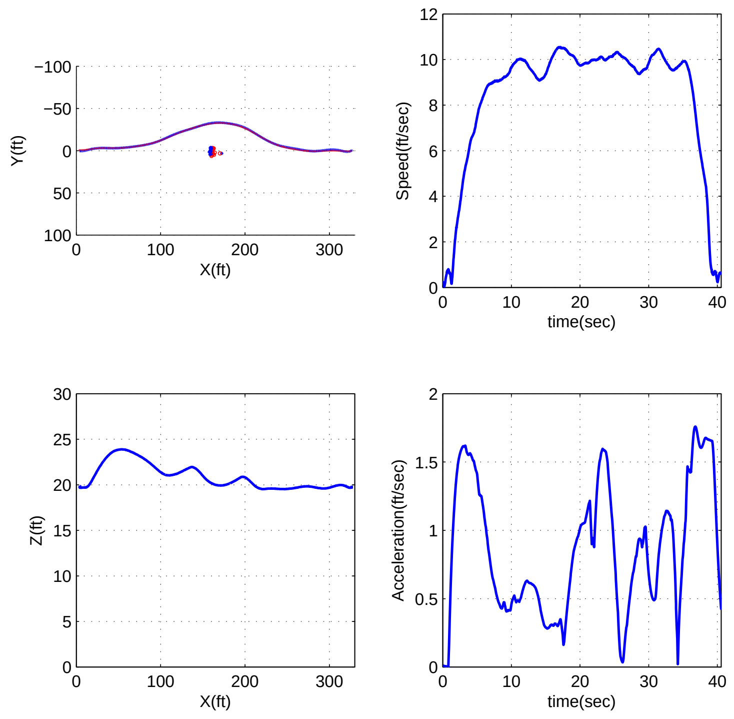


(a) free terminal acceleration

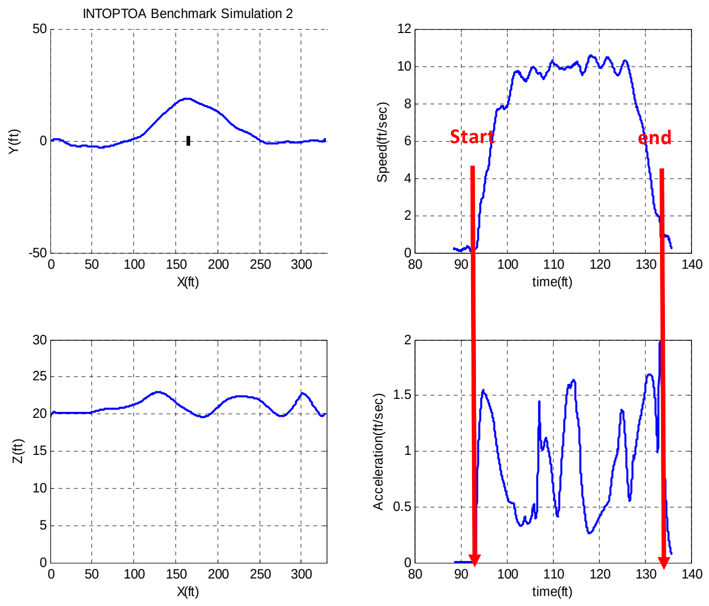


(b) zero terminal acceleration

Figure 67: Benchmark simulation case 1

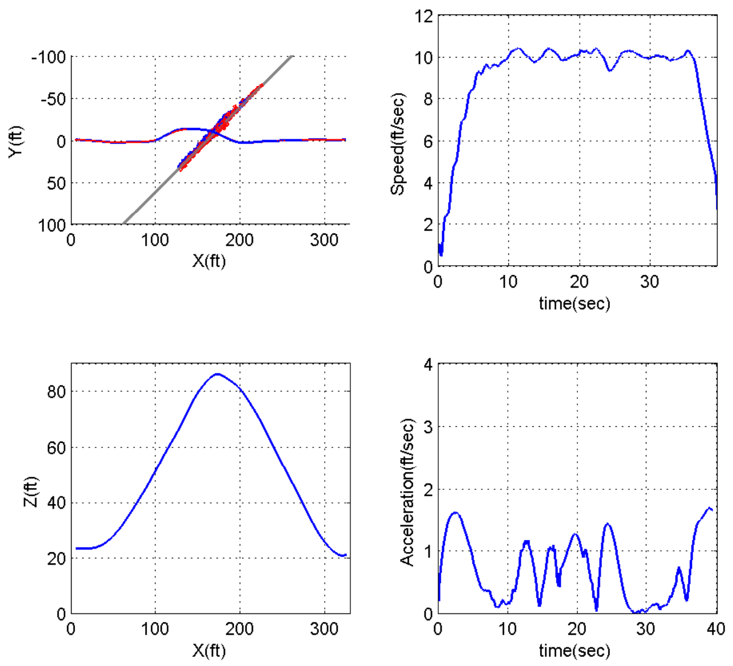


(a) free terminal acceleration

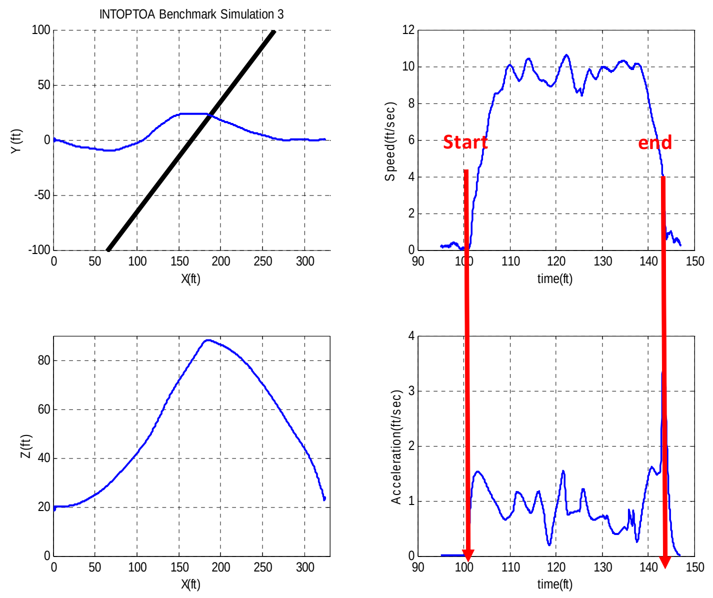


(b) zero terminal acceleration

Figure 68: Benchmark simulation case 2

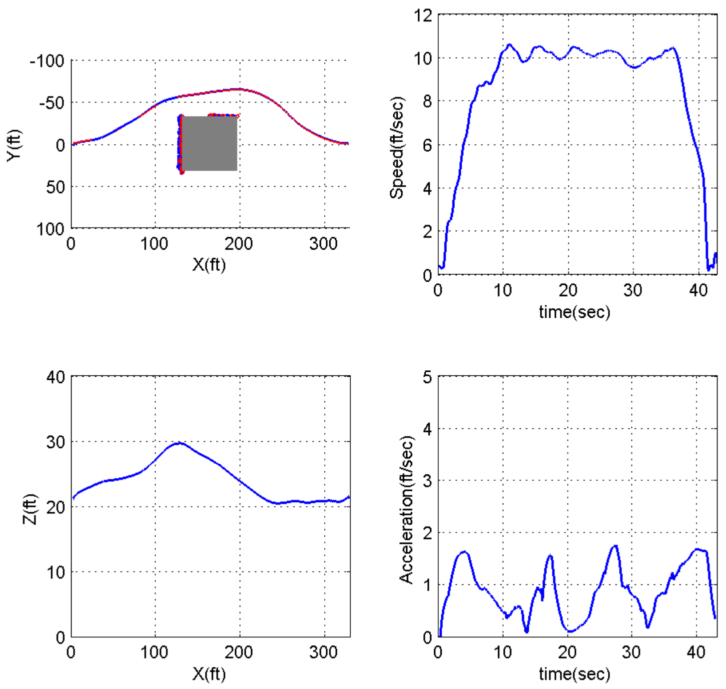


(a) free terminal acceleration

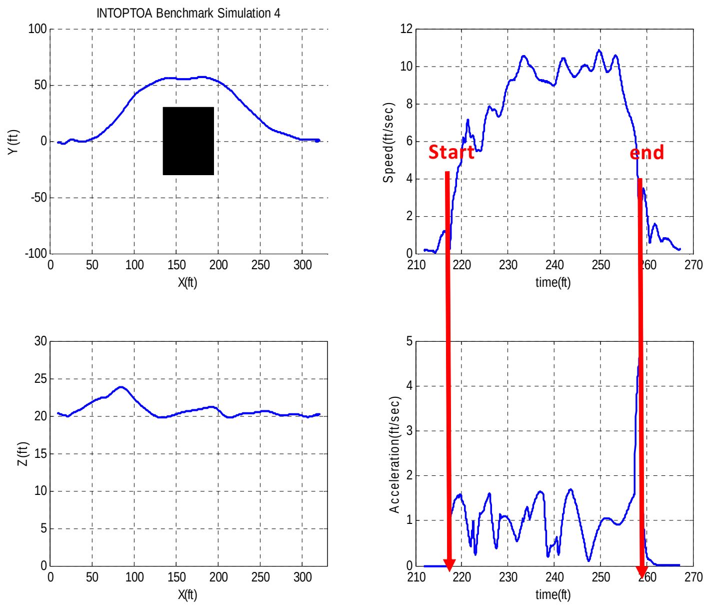


(b) zero terminal acceleration

Figure 69: Benchmark simulation case 3

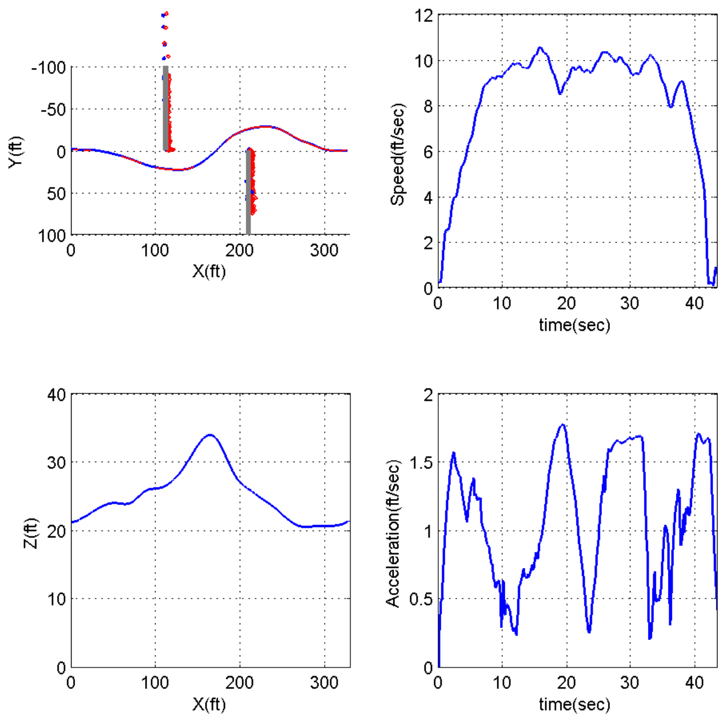


(a) free terminal acceleration

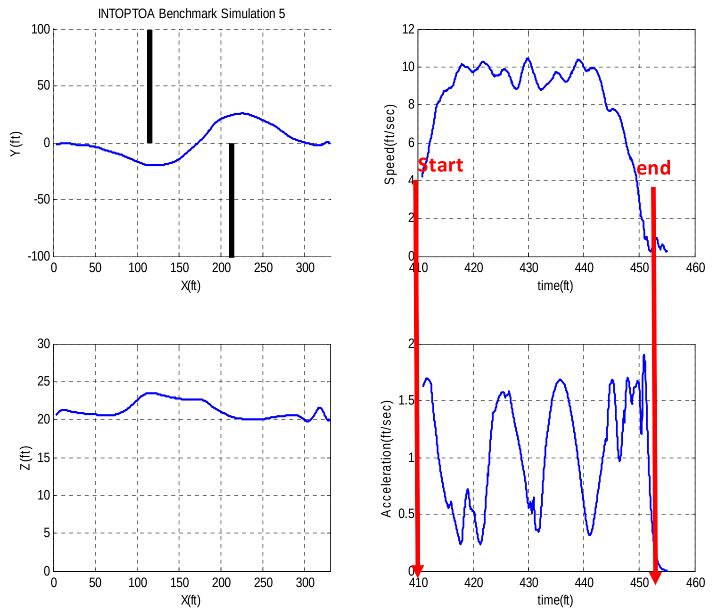


(b) zero terminal acceleration

Figure 70: Benchmark simulation case 4

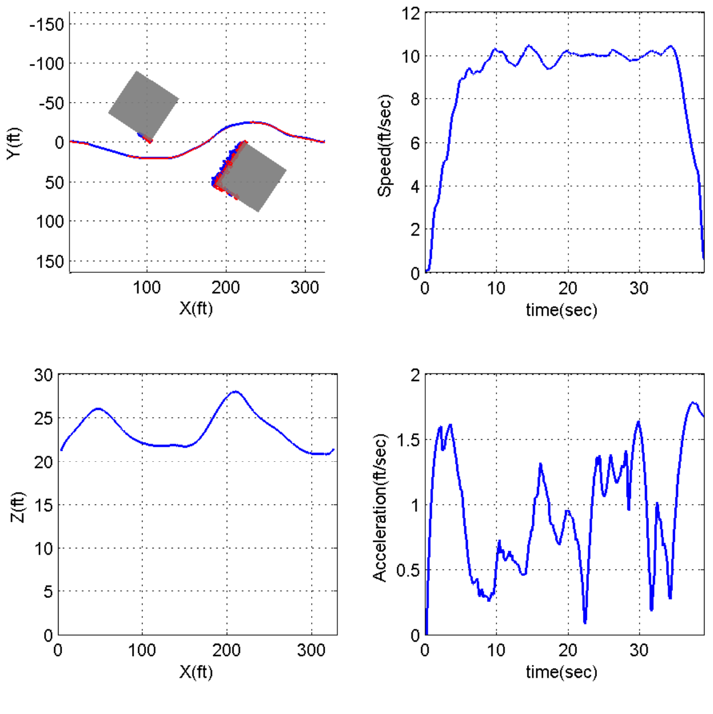


(a) free terminal acceleration

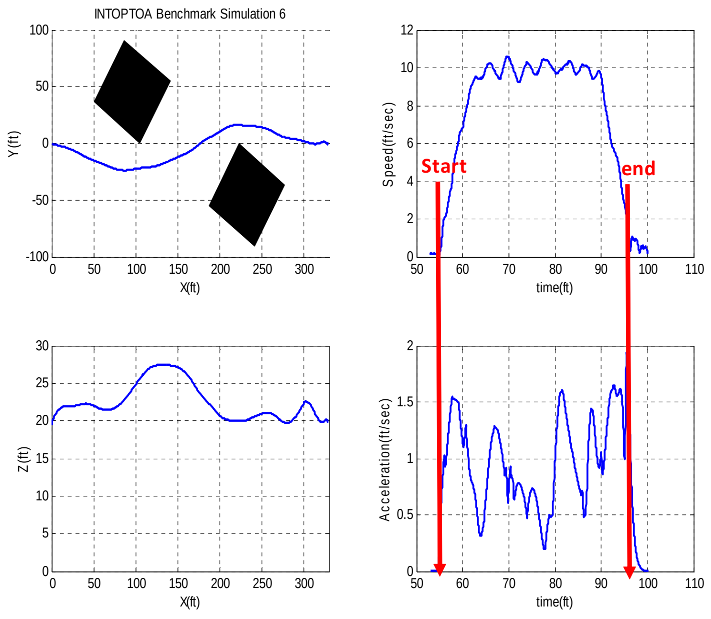


(b) zero terminal acceleration

Figure 71: Benchmark simulation case 5



(a) free terminal acceleration



(b) zero terminal acceleration

Figure 72: Benchmark simulation case 6

6.3 Benchmark Flight Tests

INTOPTOA worked well in the benchmark flight tests (see Figures 73 through 80) except for the benchmark case 3 in the first test as seen in Figure 73. In that case, INTOPTOA failed to maneuver the vehicle as expected, going around the wall instead of going over. However, notwithstanding the failure, it revealed *the inherent weakness of pure local trajectory generation using incomplete knowledge of the actual obstacle geometry.* In the flight test, the sensor range was set to $200ft$. However, the pretest simulation used $800ft$, thus allowing for the detection of a larger portion of the wall at the beginning. As can be seen in the simulation result of the benchmark 3 case in Figure 69, an earlier detection of a large portion of an unknown large obstacle could result in a successful avoidance. In addition, analysis revealed that the time delays in transfer of the sensor data and the vehicle attitude errors could also affect the result of the benchmark 3 case, and those adverse effects were confirmed in simulation as seen in Figure 81. The unsuccessful result of the first flight test of benchmark case 3 required a determination logic of the sensor measurement saturation in the lateral direction for large and long obstacles as in the benchmark case 3. The initial solution generation logic for the RH trajectory optimization was modified to take into account the fact that the detected obstacle could be larger than what the sensor could detect at a given instant when the laser returns from the sensor cover the entire field of view in lateral and vertical directions. As a result of this modification for the initial solution, the second flight test for the benchmark case 3 succeeded in going over the wall, as seen in Figure 77.

Table 5: Summary of INTOPTOA flight test results

Cases	Obstacle Type	Baseline	Sim. [1]	FT1 [2]	FT2 [3]
case 1	Out and back	78.8	80.0	-[4]	-
case 2	Point	39.3	40.0	-	-
case 3	Wall	39.3	40.0	72.0	41.0
case 4	Cube	42.1	42.0	41.0	42.0
case 5	Wall baffle	41.7	42.0	42.0	44.0
case 6	Cube baffle	39.8	40.0	39.0	40.0

[1] Simulation

[2] First benchmark flight test (Nov 22, 2011)

[3] Second benchmark flight test (Dec 16, 2011)

[4] Not tested

* Flight test results are rounded

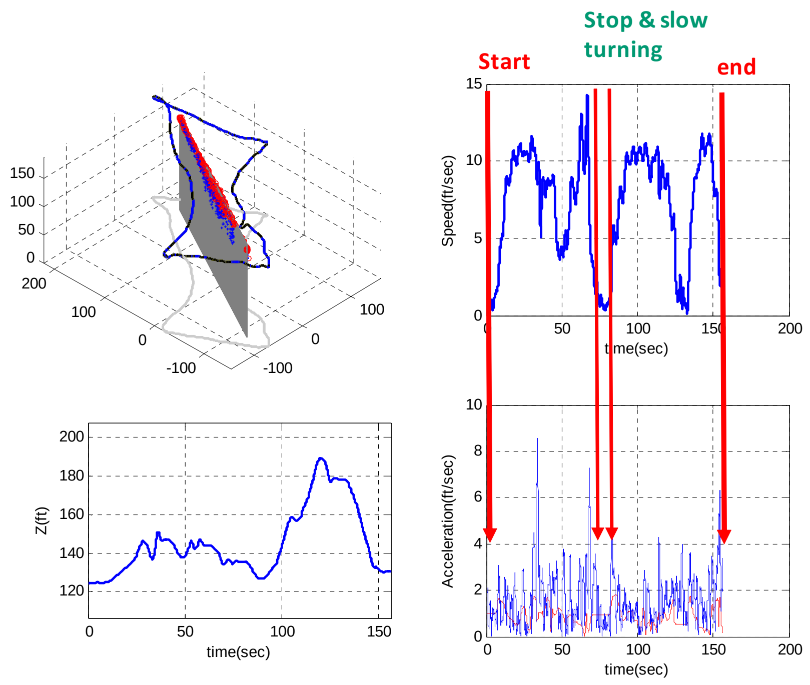


Figure 73: 1st benchmark flight test case 3: dotted marks on the obstacle are laser hits.

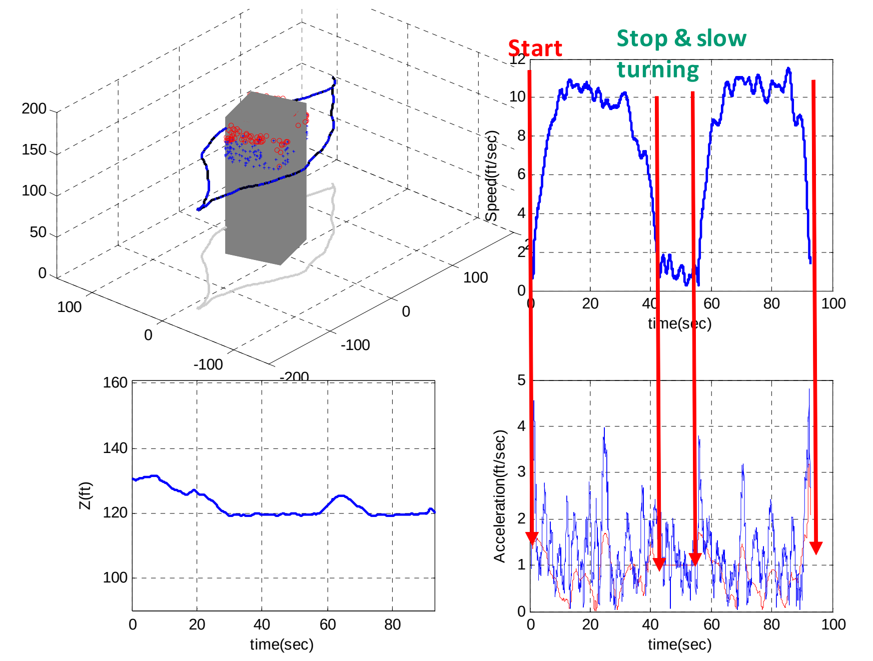


Figure 74: 1st benchmark flight test case 4

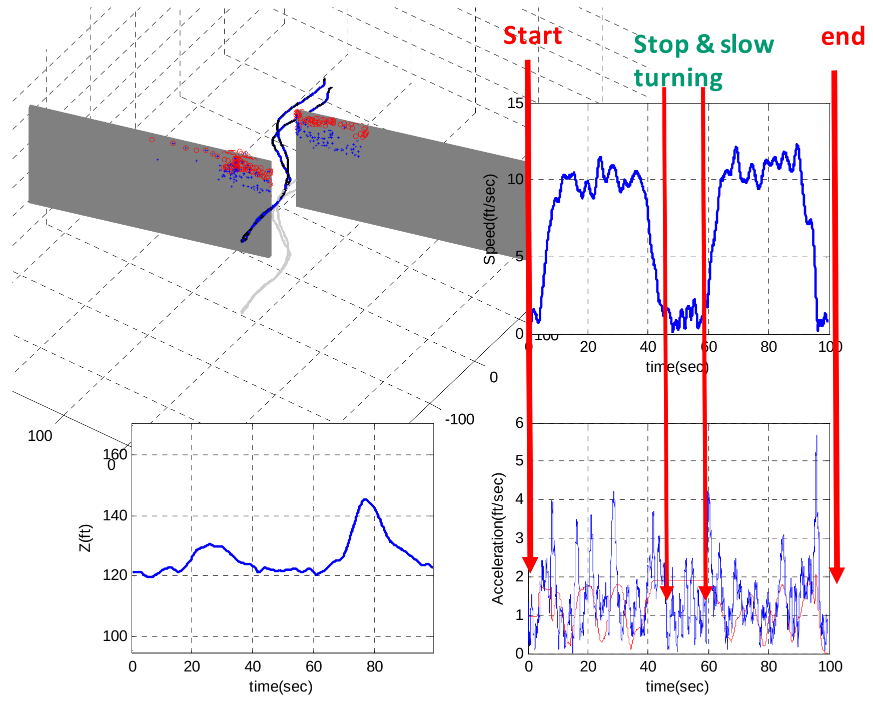


Figure 75: 1st benchmark flight test case 5

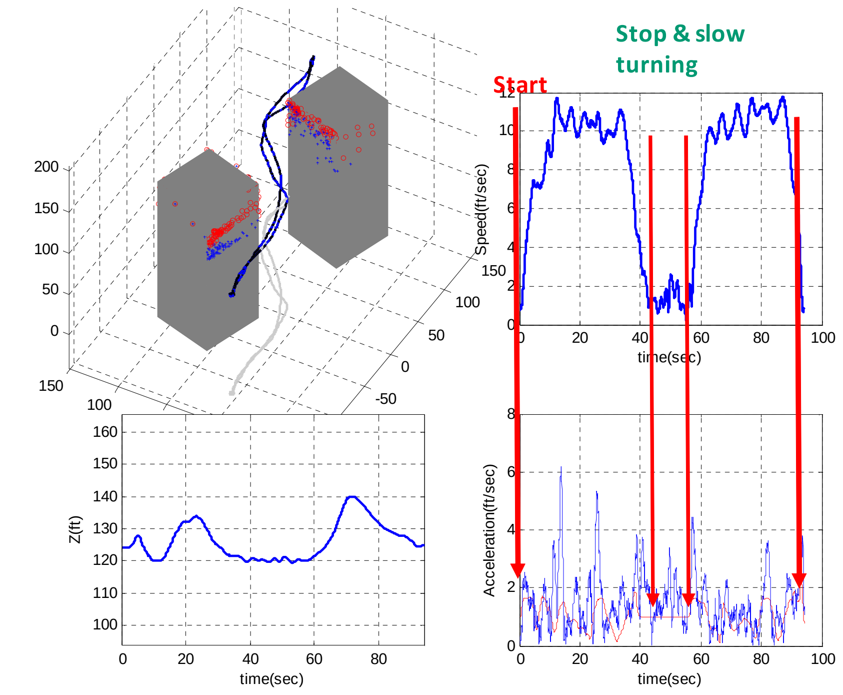


Figure 76: 1st benchmark flight test case 6

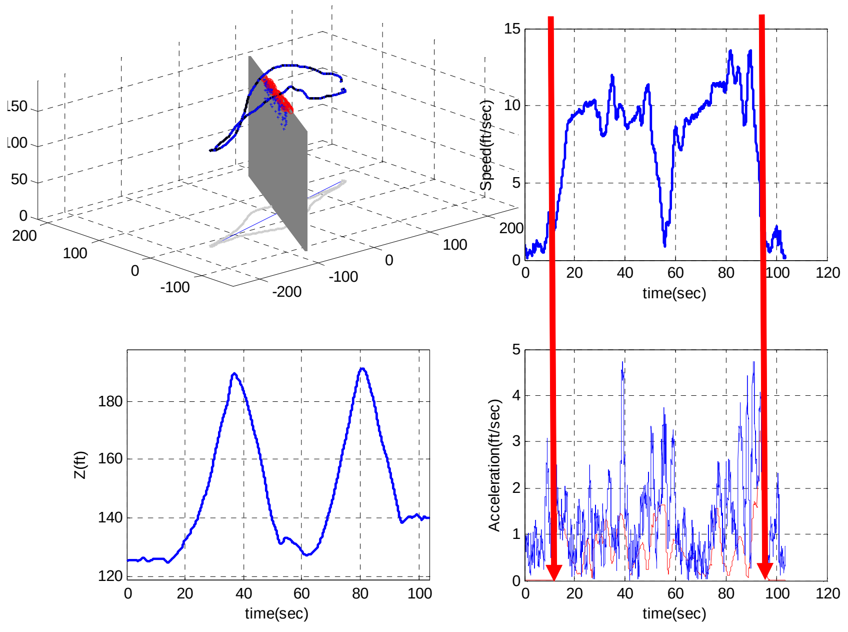


Figure 77: 2nd benchmark flight test case 3

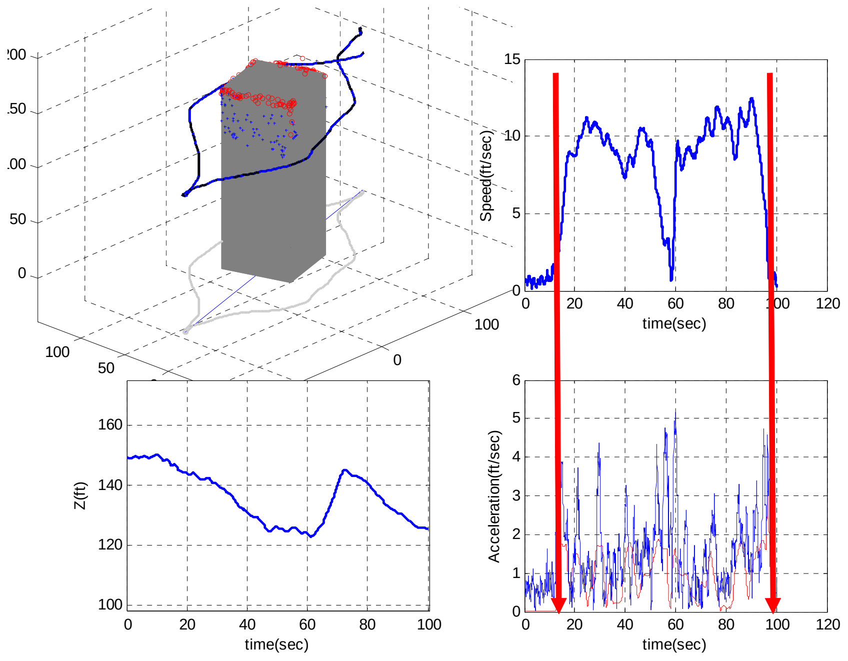


Figure 78: 2nd benchmark flight test case 4

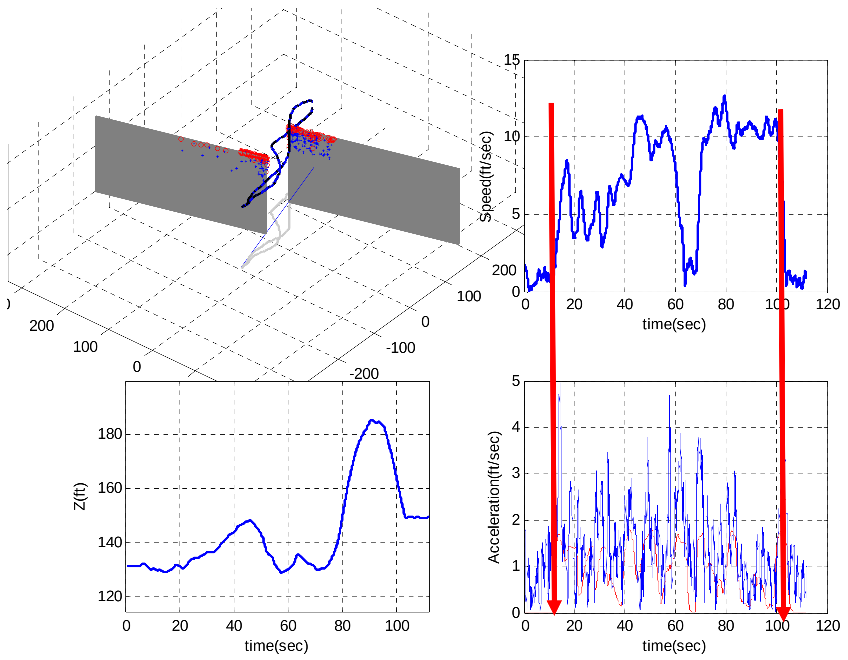


Figure 79: 2nd benchmark flight test case 5

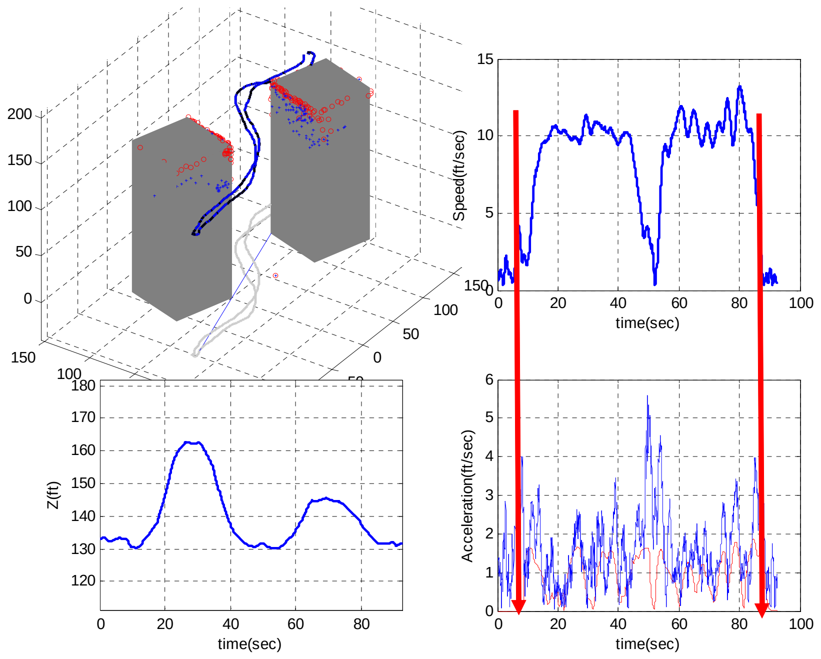


Figure 80: 2nd benchmark flight test case 6

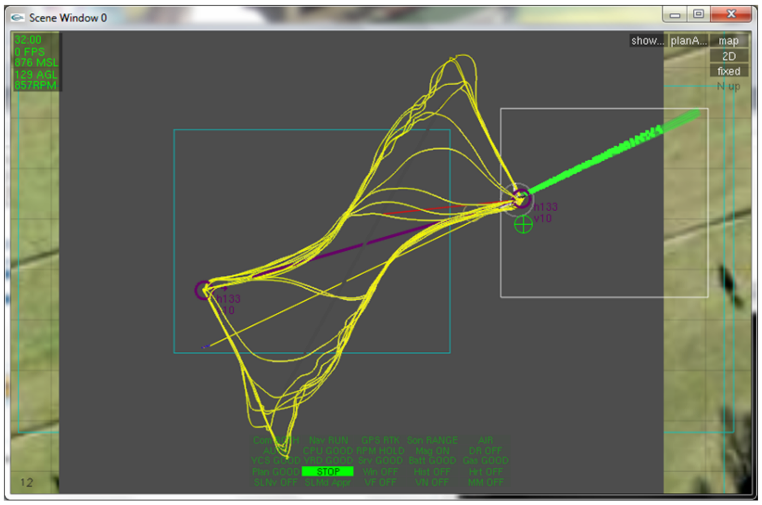


Figure 81: Benchmark case 3 simulation after the first flight test: 10 consecutive trials of back and forth movement over the wall are simulated with 200ft sensor range, 0.2sec sensor signal delay, and yaw-roll attitude coupling.

6.4 Sensor in The Loop Flight Test

After the benchmark tests at the second flight test, the sensor in the loop avoidance for a tree of about $50ft$ height was attempted. Figure 82 shows an instance of avoidance maneuver during the test. For safety, the base flight altitude was set to $60ft$ and the clearance was chosen as $50ft$. The sensor could detect the actual height and geometry of the tree, roughly L:$60ft$×W:$60ft$×H:$50ft$ as seen in Figures 83 and 84. With the initial position of the vehicle along the center-line of the obstacle, a lateral avoidance maneuver with the specified clearance of $50ft$ required the vehicle to maneuver with a greater lateral deviation from the straight path to the target waypoint compared to the vertical deviation, thus INTOPTOA relevantly generated vertical avoidances, as seen in Figures 83 and 84.

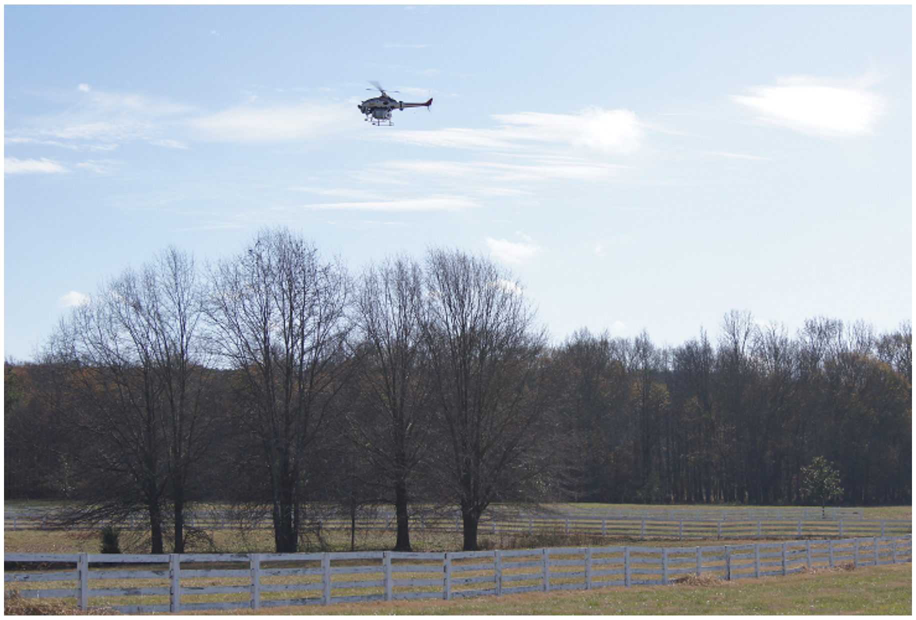


Figure 82: UAV test-bed is avoiding the tree obstacle in the flight test.

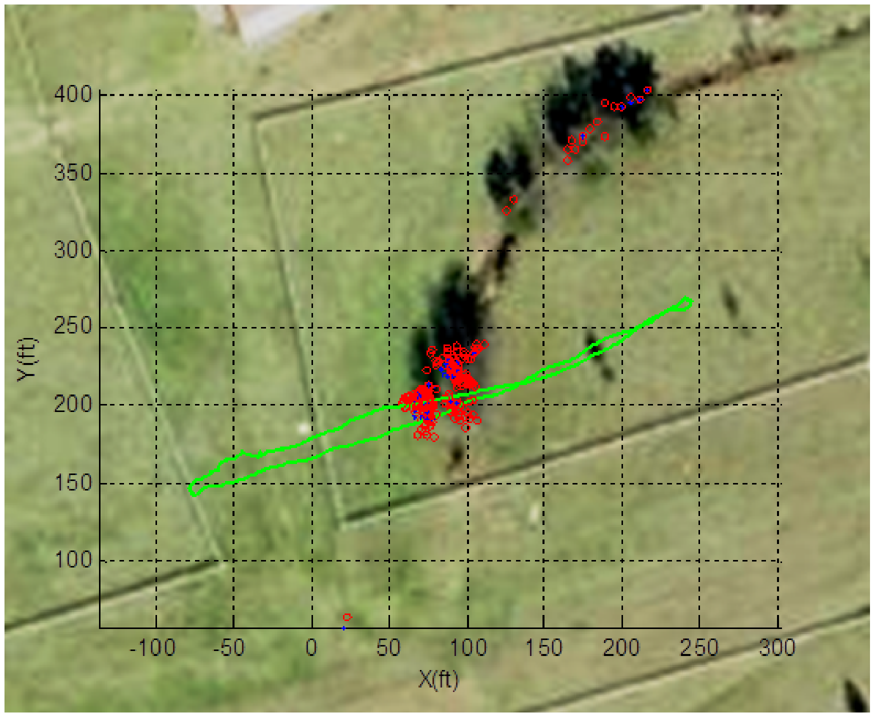


Figure 83: Actual sensor in the loop test over a tree: Red dots indicate the laser hits on trees.

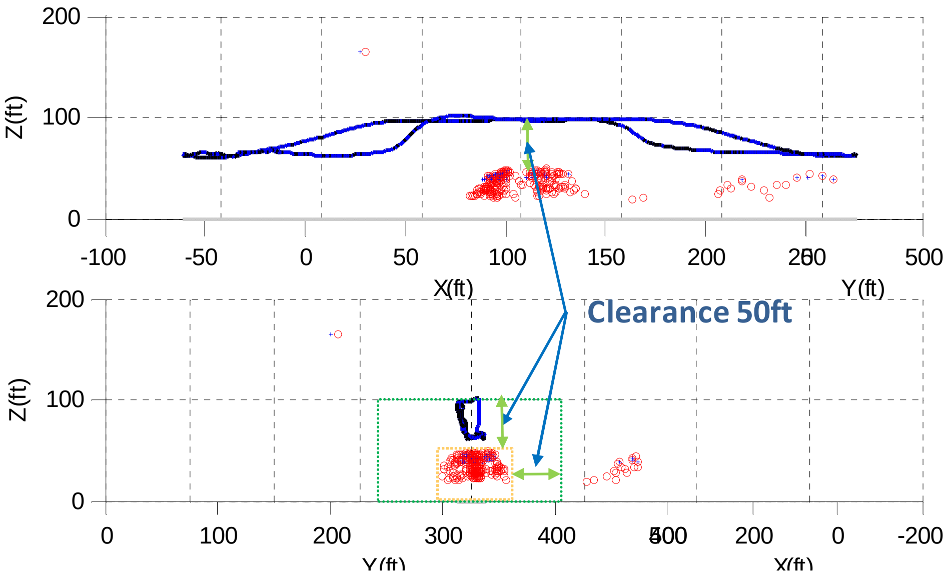


Figure 84: Projected views: the vertical avoidance was relevant to the obstacle geometry for the set clearance.

6.5 Flight Demonstration

The INTOPTOA was tested and demonstrated at the McKenna MOUT site in Fort Benning, Georgia, in January 2012. The UAV test-bed flew the closed-circuit course on the site with two different speed sets, $15ft/s$ and $25ft/s$. The clearances were chosen as $60ft$ minimum altitude and $50ft$ minimum relative distance to obstacles. A tree with a height of roughly $50ft$ on the south-west part of the course was selected as the primary avoidance target and a group of small trees lower than $30ft$ on the north-east part of the course were selected as the secondary target. The test was done by sequentially conducting the back and forth avoidance trials on the primary obstacle with $15ft/s$ speed, the closed-course flight at $15ft/s$, and the closed-course flight at $25ft/s$. The flight test results are presented in Figures 85 through 88 for the $15ft/s$ flight and in Figures 89 through 92 for the $25ft/s$ flight.

Overall, the INTOPTOA algorithm guided the UAV successfully during the flight test demonstration. Avoidance of the primary target coincided with the expected result from simulations, referring to the south-west portion of trajectories of the flight test and the simulation compared in Figure 93. Guidance to the waypoints over the no-obstacle region was normal, keeping the clearance height of $60ft$ above the terrain with a straight path to the waypoints (see Figures 87 and 91). The relative clearance was maintained above the minimum of $50ft$ from the measured obstacle geometries (see Figures 88 and 92), except for a short duration of violation around $200sec$ in the $25ft/s$ trial (see Figure 92) which was caused by an abnormal sensor measurement around the region.

However, during the $15ft/s$ speed case, two issues associated with the actual implementation of the INTOPTOA algorithm was detected. First, the INTOPTOA algorithm caused unnecessary stopping twice at two different way points, which was later found to be associated with the faulty combination of a safety stop logic and the waypoint transition logic. Second, an abrupt change in the command trajectory occurred during the avoidance of the secondary target. Subsequent analysis and simulations revealed that an unremoved obsolete logic in the algorithm caused the abnormal command that veered the vehicle from the proper direction unnecessarily, though the correct path was recovered subsequently. See the top-right portion of the flight path in Figure 93(a) which shows the overlapped trajectories of the flight test and the multi-pass simulations. One pass of the multi-pass simulation and the abnormal portion of the flight test path are similar to each other. Such abnormal changes of path happened regardless of the presence of an obstacle in the simulation trajectories as seen in the bottom-right portion of the simulation trajectory in Figure 93(a).

The test with $25ft/s$ speed was successful in avoiding both obstacles except that the sensor was presumed hit by direct sun light or the strong reflected light, causing the false measurement around the secondary target and pushing the vehicle to the left of the expected path over that region (see Figure 94). The false laser hit points were distributed along the line of sight to the sun near the secondary target as seen in Figure 94. In fact, this is not an unusual but an inherent problem when using the laser-type sensors. It should be noted that laser sensors have problems related to specular reflections of laser, less sensitivity on glossy surfaces, and beam bouncing, all of which can cause errors in the measurements.

The problems associated with the implementation of the INTOPTOA algorithm discovered from the flight test, except for the false measurement, were later fixed and verified in simulations. Figure 95 shows a result of multi-pass simulation with the modified version, demonstrating that INTOPTOA could produce consistent guidance trajectories for the same McKenna flight test conditions.

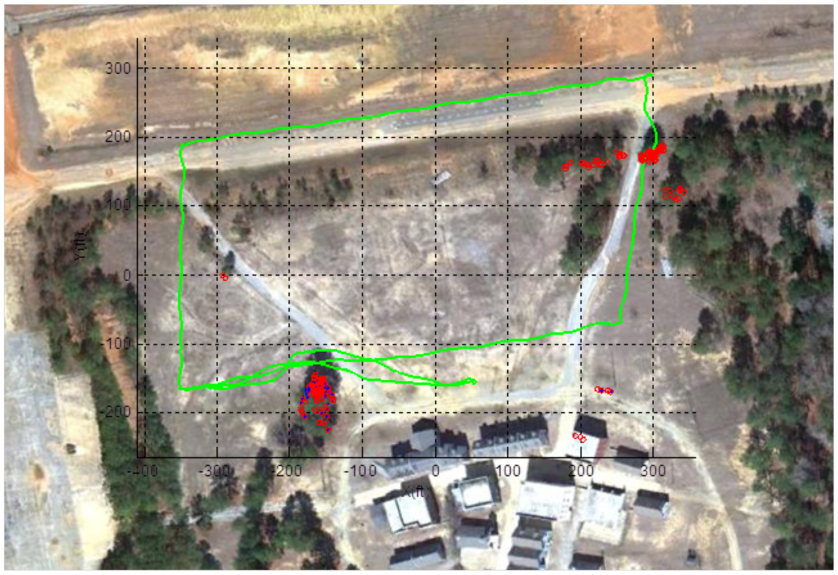


Figure 85: McKenna MOUT site flight test trajectory ($15ft/s$): Top projection.

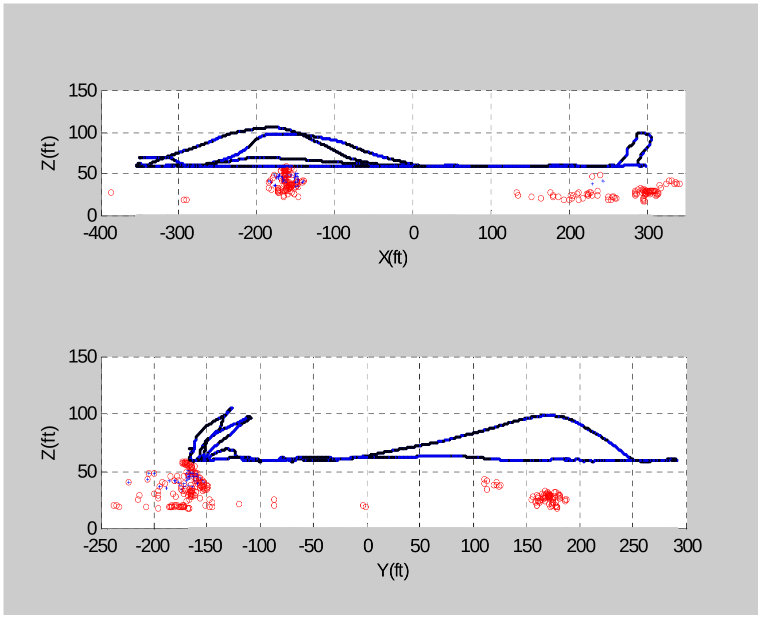


Figure 86: McKenna MOUT site flight test trajectory ($15ft/s$): Side projections.

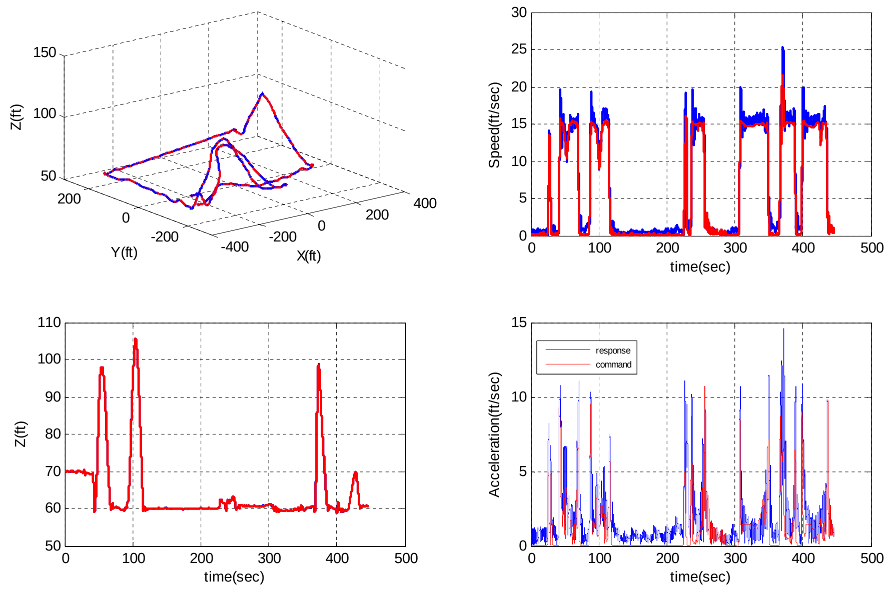


Figure 87: McKenna MOUT site flight test trajectory ($15 ft/s$): Altitude, velocity, and accelerations.

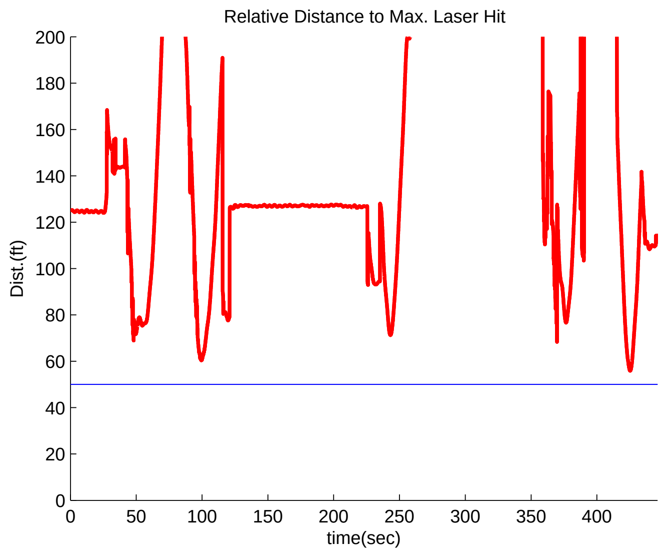


Figure 88: McKenna MOUT site flight test trajectory ($15 ft/s$): Relative clearance to measured maximum height point of obstacle.

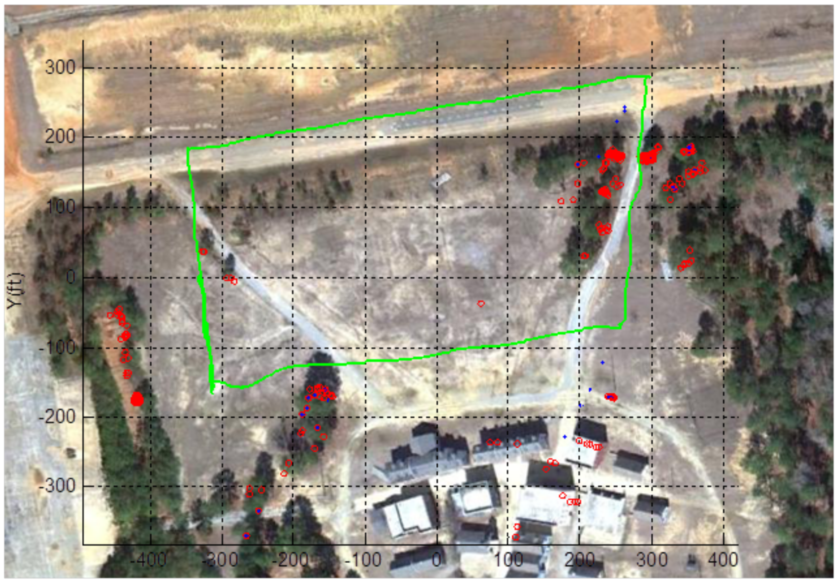


Figure 89: McKenna MOUT site flight test trajectory ($25ft/s$): Top projection.

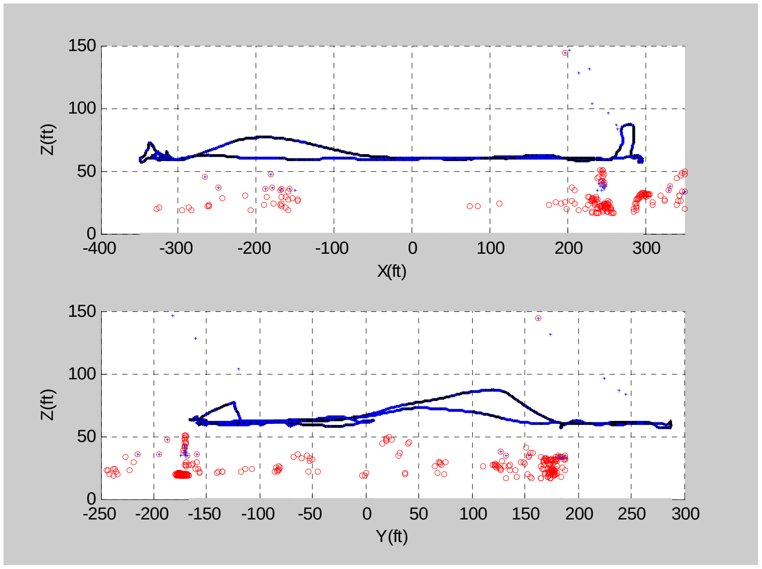


Figure 90: McKenna MOUT site flight test trajectory ($25ft/s$): Side projections.

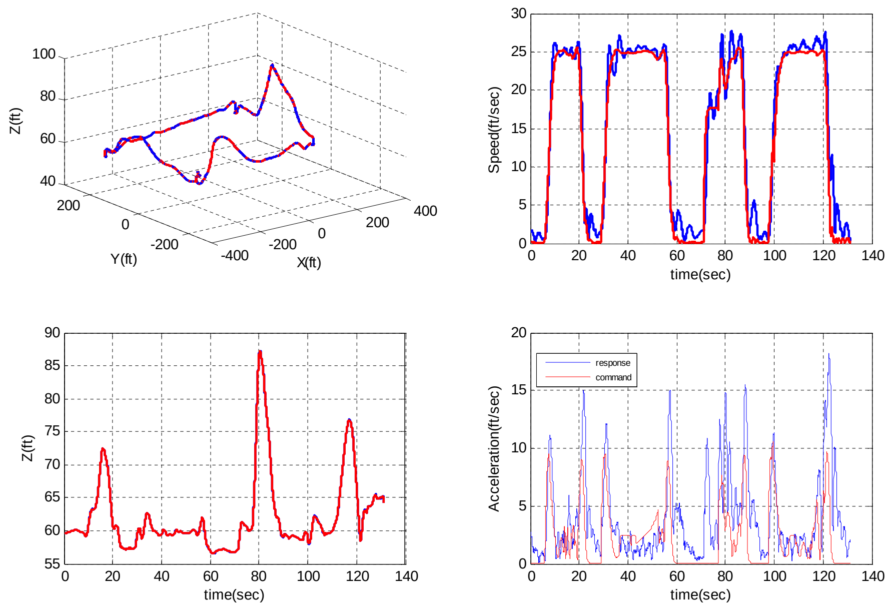


Figure 91: McKenna MOUT site flight test trajectory ($25ft/s$): Altitude, velocity, and acceleration.

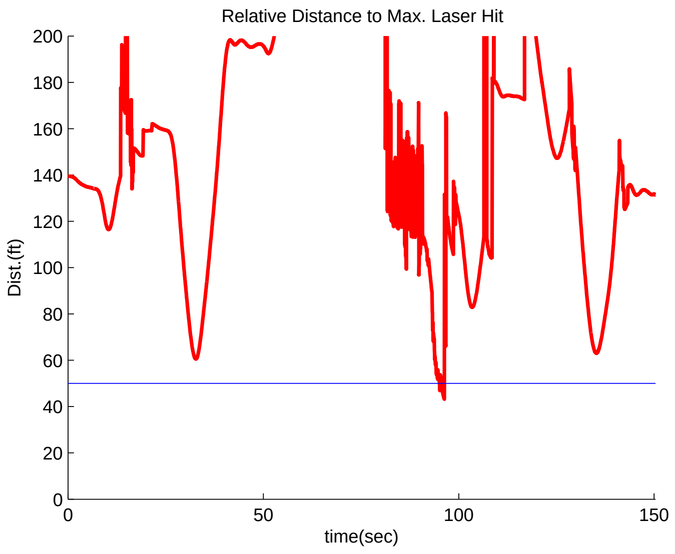
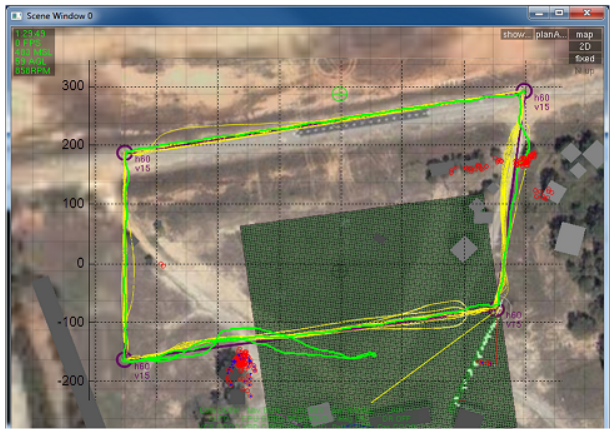
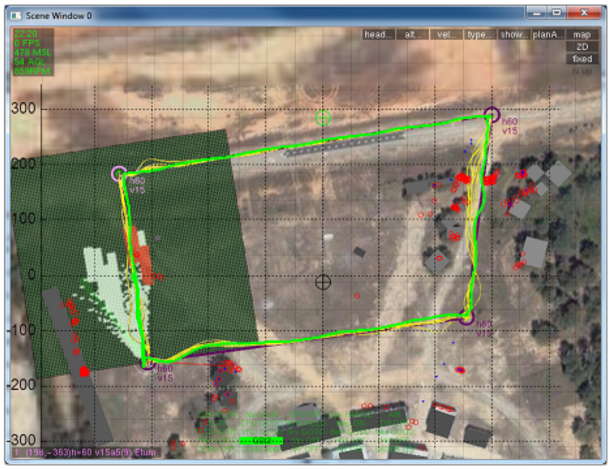


Figure 92: McKenna MOUT site flight test trajectory ($25ft/s$): Relative clearance to measured maximum height point of obstacle.



(a) $15 ft/s$



(b) $25 ft/s$

Figure 93: Comparison of flight test trajectory to multi-pass simulation

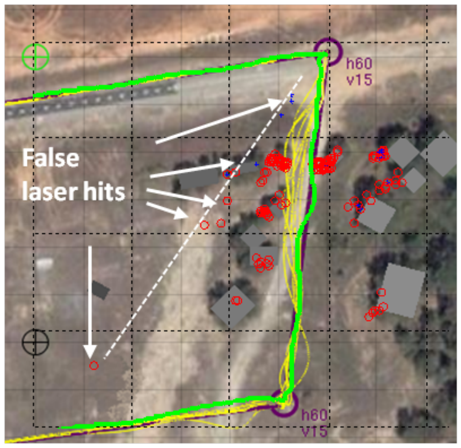
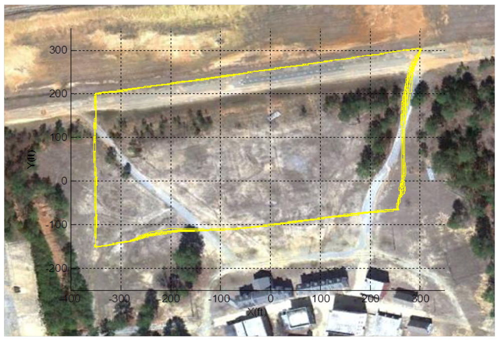
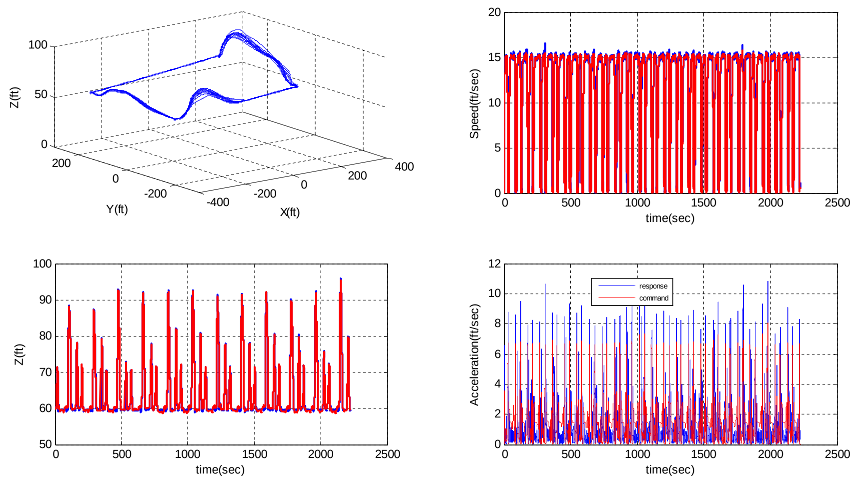


Figure 94: False laser hits are scattered in the line



(a) x-y trajectories



(b) Heights, velocity, and acceleration

Figure 95: Multi-pass simulation for the McKenna MOUT site flight test case of $15ft/s$

6.6 Summary

Test and evaluation on the performance of INTOPTOA has been done in simulations and flight tests. Benchmark tests and the demonstration at McKenna MOUT site show that the RH trajectory optimization module can generate the command trajectory in real time, that sensor integration to construct obstacle grid works normally, and that INTOPTOA can provide continuous obstacle field navigation capability with the real-time optimization technique, as long as the obstacle field measurement is normal.

CHAPTER VII

CONCLUSIONS

7.1 Summary

This thesis presents an integrated framework suitable for rotary-wing unmanned aerial vehicles to conduct low-altitude operations in partially or completely unknown environments like an urban area. The developed framework is based on the receding horizon trajectory optimization in conjunction with a fast global path searching.

The concept of the receding horizon trajectory optimization is similar to the well-known receding horizon control (RHC) or model predictive control (MPC). It solves a series of trajectory optimization problems formulated as finite horizon optimal control problems at the current vehicle state, and the open-loop optimal trajectory is used as the command input for the vehicle. The command inputs are continuously interpolated from the previously computed optimal trajectory at the current vehicle position. When the new solution is obtained, the trajectory for interpolation is replaced with the new optimal trajectory, and this process is repeated till the vehicle reaches the target point. Unlike other receding horizon controls focusing on the generation of a direct control input to the vehicle, the RH trajectory optimization of this thesis computes the local optimal trajectory to be followed by the vehicle controller. So, the overall architecture of the trajectory planning for obstacle avoidance can be considered as a *two-layer architecture* of trajectory planning.

RHC or MPC is the control technique based on the finite horizon optimization. It has been applied to large multi-variable process control or plant control in the petro-chemical and process control industries of a few decades ago, mainly due to the economic consideration that requires a plant to be operated within limits. Since then, this control approach has received wide attention in the broader field of control theory and other applications because of the benefits of the approach: it can naturally handle multi-variable systems, it can systematically take actuator limitations into account, and it allows a system to operate closer to its constraints, thus often resulting in better performance. Based on these advantages, RHC has recently been applied to trajectory planning problems of fast dynamical systems. Obstacle avoidance of UAVs using receding horizon optimization is one such effort. The heart of the receding horizon optimization and the main propelling force of expanding its applications to the control of fast dynamical systems is an evolving real-time optimization technique.

This thesis used Nonlinear Trajectory Generation (NTG) as the real-time optimization solver and integrated it into the trajectory planning framework of a UAV helicopter. NTG is a direct solver which uses the spline approximation of the output of the flat system and converts the optimal control problem of the trajectory optimization into a nonlinear programming (NLP). The converted NLP is solved by a sequential quadratic programming-based NLP solver, NPSOL. NTG can compute a trajectory optimization problem almost in real time, in less than at most hundreds of millisecond, if the horizon of the trajectory is finite and a low-order spline is enough for the approximation of the trajectory.

During the initial phase of the development of the framework, this thesis explored the time-optimal avoidance for a single obstacle, then for multiple obstacles represented as rectangles, and for multiple layers of arbitrary shape manifolds encompassing obstacles in it. The time-optimal avoidance approach for a single obstacle and the real-time optimization module were integrated to run on the onboard computer of the UAV test-bed and were successfully flight-tested on virtual obstacles. Perhaps the most important knowledge acquired from the first flight test was the discovery of the vehicle power saturation during the avoidance climb maneuver and the fact that an overestimated value of climb rate limit can produce an infeasible command trajectory. As an effort to resolve this issue, a limit detection logic for climb rate was proposed and was evaluated in simulation.

Along with this safety issue associated with vertical obstacle avoidance, the basic approach of trajectory optimization was fundamentally changed from the single optimization before avoidance maneuver to the continuous optimization until the vehicle reaches the destination. The idea of receding horizon trajectory optimization was incorporated into the trajectory planning framework of this thesis, and the framework was interfaced with an actual sensor, a LIDAR, to measure the geometry of obstacles and terrain. Finally, the receding horizon trajectory optimization scheme was extended to the three-dimensional trajectory optimization using a moving local obstacle grid map constructed from the point cloud data from LIDAR. In addition, to increase the *completeness* of the local trajectory generation and to provide an appropriate initial guess to the real-time optimizer, a coarse global path searching algorithm was added to the framework.

The global path searching algorithm is based on a graph searching by dynamic programming over the simplified obstacle cuboids, which can either be constructed online from the current obstacle grid being used in the receding horizon trajectory optimization or be transferred to the framework from an external source. A blob detection algorithm is applied to detect and extract obstacle cuboids from the obstacle grid, and the framework manages the cuboids to be merged or inserted into the existing database.

The developed framework was implemented in the Georgia Tech UAV Simulation Tool (GUST) and was embedded in the onboard computer of the UAV test-bed at Georgia Tech. An important aspect of the implementation was realizing the actual scheme of the receding horizon trajectory optimization, that is, *using the previous optimal trajectory until the solution is updated*. This scheme could be realized by implementing the trajectory optimization module as a multi-threaded routine. Simulations and flight tests had been done throughout the development of the framework, and a total of seven flight tests were conducted for the basic vertical avoidance, benchmark tests, and the final demonstration of three-dimensional avoidance. Through such evaluation efforts, the framework was continuously improved to its envisioned purpose, *automatic obstacle avoidance in an unknown environment*, thus widening the feasibility of future application to collaborative low-altitude missions in an urban environment.

7.2 Conclusions

From the results obtained in this study, the following general conclusions can be drawn:

- *In-flight trajectory re-planning for obstacle avoidance by RH trajectory optimization becomes practical when it is implemented with a real-time solver and the multi-threaded computation of an optimization process.* Without a real-time solver, in-flight trajectory optimization becomes impractical for a fast dynamical system like a UAV. It is obvious that a fast dynamical system requires fast updates of trajectory, especially for obstacle field navigation in an unknown territory. If the vehicle controller is a type of trajectory follower, inconsistent trajectory commands due to a computational delay in optimization may directly affect the safety of the vehicle. Multi-threading the optimization process is an effective way to generate consistent trajectory commands.

- *RH trajectory optimization only considers the finite horizon of trajectory for optimization. Thus, the resultant trajectory does not necessarily guarantee an optimality in a global sense.* This fact is confirmed by the case study on the time-optimal avoidance of multiple layers of obstacles in Chapter 3, as the local time-optimal trajectories obtained from subsequent obstacle detection does not form a true time-optimal solution that can be obtained at once with the complete knowledge of the obstacle field. Another disadvantage of local trajectory optimization can be found in the benchmark flight test for case 3. If the vehicle cannot detect the entire obstacle geometry at a distance, the resultant flight path can end up with an undesired result. One way of resolving this weakness of local trajectory optimization is incooperating a fast global path search algorithm using a roughly represented obstacle field beyond the finite horizon of RH trajectory optimization.

- *RH trajectory optimization is an effective method of trajectory planning subjected to multiple constraints including dynamics constraints.* However, its usefulness largely depends on a robust convergence of the solutions in real time. If the real-time computation is the main interest, NTG is a suitable solver because it is fast if a system can be represented as a flat system. However, it is inherently inapplicable to a non-flat system, and it has difficulty in finding a converged solution if the initial guess is infeasible. Providing a feasible initial guess to NTG is critical to the convergence of solution as well as computation time.

- *The previous successful solution can be used as an initial guess for the current optimization, and it can enhance the convergence of the solutions as well as the computation time.* This thesis employs this technique in actual implementation of the algorithm, so the overall computation time is reduced. However, the resultant trajectory tends to maintain the previously computed trajectory unnecessarily even though the avoidance condition is changed, making the optimal trajectory different from the previous one. The global path search algorithm can be used to provide a feasible initial guess for the optimization as well as to guide the local optimal trajectory to the global path. In the actual implementation of the algorithm, the initial guess is sampled from a global path once it is available.

- *Differences in terminal constraints in the formulation of RH trajectory optimization may not affect the overall avoidance trajectory, depending on the length of the horizon.* This aspect of RH trajectory optimization is seen in the benchmark case simulations with different terminal conditions, i.e., zero or free terminal acceleration. As presented in Chapter 6, the overall trajectories of benchmark cases with different terminal acceleration conditions do not show notable differences. This may be a result of the RH trajectory optimization: only a small portion of the previously computed trajectory is used as the command to the vehicle before the command trajectory is updated by the new optimal solution; hence, the initial part of the optimal trajectory cannot be significantly influenced by the terminal constraints, especially if the terminal position of the current optimal path is sufficiently far from the current position.

- *RH trajectory optimization provides local suboptimal solutions.* Most of MPCs or RHCs based on direct optimization methods use suboptimal solutions for their controls or trajectory planning. Apparently, there is a trade-off between the computation time and the accuracy of the optimal solution. If an optimization problem focuses on an exact optimal solution, solvers based on direct methods might be insufficient. A combination with other accurate optimization techniques based on indirect methods may be needed.

- *Minimization of the integrated quadratic distance to the target point can be a suitable objective for a 3D trajectory optimization for obstacle avoidance.* With an appropriate set of weighting factors of the integrated quadratic distance along each axis of the coordinate frame, it may indirectly produce a near time-optimal trajectory or a trajectory with minimal deviation from the straight path to target point. As proven in the benchmark simulations and the flight tests in Chapter 6, the RH trajectory optimization with the selected cost function is capable of producing an avoidance trajectory closer to the theoretical solutions that assume complete knowledge of the obstacles.

- *Fast and wide measurement of the obstacle field is a basic requirement for the robust operation of the proposed framework.* The limitations to obstacle measurement such as short detection range, narrow scanning volume, and slow 3D search can impact the overall performance of the trajectory planning. This is observed in the first flight test for the benchmark case 3. The disadvantages of the obstacle detection scheme used in this study, i.e., short detection range and slow yawing to cover the lateral region, can lead the framework to fail to find the proper avoidance path. The obstacle detection scheme used in this study may become more sensitive to the requirement of fast and accurate sensor measurements when the vehicle speed is further increased.

- *Overestimated vehicle performance limit can be a significant safety issue during avoidance.* Especially for vertical maneuver of a rotor-craft, this thesis shows that the use of excessive limits of climb rate in the optimization may result in an unsafe situation during vertical avoidance. An appropriate method of detecting the performance limit, like the climb limit detection logic proposed in this thesis is important for the safety of avoidance as well as for increasing the agility of avoidance.

7.3 Recommendations for Future Works

7.3.1 Convergence of optimal solution and back-ups for safety

This thesis shows that real-time receding horizon optimization is useful in trajectory planning for obstacle avoidance, but this usefulness presumes the optimal solution is attainable in a short computation time in any situation. However, in reality, issues such as the existence of an optimal solution, the computation time and the convergence of the numerical optimization always exist and should be carefully accounted for in the actual implementation of an optimization-based trajectory planning framework. Especially in the obstacle avoidance problem, the sudden appearance of an unforeseen obstacle at a close distance is a possible situation in practice, and it often endangers the convergence of the solution because the problem configuration could easily turn out to be infeasible to the hard constraints. For example, if an obstacle suddenly appeared within the clearance distance, the optimal solver will not be able to come up with a new optimal solution satisfying the constraint on the clearance. In such situations, it might be necessary to relax the hard constraints or to switch to an alternative reactive avoidance algorithm to prevent a collision.

7.3.2 Hard constraint protection

Even though velocity and acceleration limits are formulated as hard constraints in trajectory optimization, especially when the vehicle is operating at limit boundaries, i.e., accelerating with maximum accelerations or flying at maximum velocity, etc., actual transient vehicle response generally can violate the limits because the role of the trajectory optimizer is to generate a reference command, not to regulate feedback error, so it may not suppress the violation quickly. In addition, the simple 1st order model of vehicle dynamics plus controller taken in the optimal problem formulation may not be sufficient to represent an accurate dynamic response in the vehicle acceleration. Thus, the open-loop nature of trajectory optimization and the imperfect representation of vehicle dynamic behavior may end up with causing transient violations of hard constraints on motion variables as the trajectory optimizer usually drives the vehicle to fly on the hard constraint limits. For example, the optimizer will produce the command trajectory requiring the maximum acceleration when it senses a large obstacle at a near distance. If the constraint limits are set to conservative values implicitly allowing some degrees of violation, the current method of optimization is enough, but a too conservative setting of the constraints usually narrows the space for a feasible solution and decreases the agility of the avoidance that would be needed in some emergency conditions. On the other hand, larger constraints on vehicle motion can increase the agility of the avoidance maneuver, but there is a possibility of further violation of the limits or the generation of a command that the aircraft simply cannot follow, such as the climb rate limit presented in Chapter 4. A method of detecting the actual limit of climb rate has been studied and evaluated in simulation; however, further studies on the hard limit protection of descent speed and the acceleration limit may be needed to increase safe avoidance capability as well as agility. One method of avoiding this is to add soft constraints as extra weighted terms in the cost function, which is similar to *a barrier function in the interior point method*, or to add a feedback logic to adjust the current reference command so as to remain within specified vehicle limits.

7.3.3 Robust sensor measurement and obstacle grid generation

A more intricate method of processing the LIDAR data is needed to establish the efficient and robust measurement of obstacles. As already described in this thesis, during the developmental flight tests involving a real sensor, the algorithms were directly affected by abnormal sensor measurements such as ambient noise detection, variation of measurement performance on natural obstacles like trees, and false laser reflections from direct sun light. Especially for a tree, the sensor sometimes showed a reduced measurement range of less than 200ft and was unable to detect the top portion of tree, thus failing to detect its actual height. Those real-world situations suggest that further study is needed on the filtering of natural factors that can influence measurement accuracy or cause malfunctions and on alternative methods which can take into account the categorized obstacle geometric characteristics, time variation of sensor measurements between samples, inherent false measurement sources, etc. The probabilistic occupancy map generation method like the certainty-assisted spatial filtering could be an alternative for this purpose

7.3.4 Need for 3D scanning laser or multi-sensor fusion

In this study, the 3D obstacle search was conducted by oscillatory yawing of vehicle attitude with vertically scanning LIDAR. Yawing motion is currently the only way to acquire the lateral field of view for a 3D obstacle field search because the framework used a 2D scanning LIDAR. In fact, the vehicle yawing, the sensor range, and the vehicle speed are closely related to the overall limit of the obstacle avoidance capability. The large frontal field of view with a high sampling rate is essential to safe avoidance trajectory generation, but the realization of it by the oscillatory yawing motion is limited by the vehicle performance, coupled with translational motion and the actuator saturation. Because the trajectory optimization of the framework only considers the translational motion of the vehicle, the oscillatory yawing motion can become a perturbation to the vehicle translation motion variables, so it can be a source of violation of hard constraints on the velocities and accelerations, and the unnecessary coupling effect on the translational motion can occur. To sum up, increasing yawing motion with high amplitude and frequency is the only way to realize a fast and wide search of an obstacle field, but it may cause actuator saturation, motion coupling, and degraded trajectory following. On the contrary, decreasing yawing motion results in slow and narrow updates of the obstacle field and can be a significant problem in dense obstacle fields. In the flight tests, because the test speed was chosen to be relatively slow and the obstacle density was low, the yawing command was set to mild yawing with $0.2Hz$ frequency and the 40-degree sweep angle.

In fact, the yawing of the vehicle has contradictory requirements. A fast and large amplitude oscillatory yawing command may be needed for 3D obstacle field detection at high speed which may become difficult for the vehicle to follow. However, high flight speed definitely requires a fast and wide sampling of the obstacle field to achieve safe obstacle avoidance. This fundamental limitation of the current approach to obtain the obstacle measurement obviously requires another method to get a fast and wide measurement of the obstacle field. An actuated sensor mount is one possible way to use the current sensor. A more fundamental solution would be to find a sensor capable of 3D scanning. The fusion of different kinds of sensors such as a camera and the laser sensor could be an alternative method.

7.3.5 3D obstacle map construction

Provided that fast acquisition of the forward obstacle field measurement is possible, then the current trajectory optimization approach can be extended to a more ambitious avoidance capability, *the avoidance through empty holes in 3D space*, such as flight under wires, bridges, branches of tall trees, etc. The current method is not fundamentally applicable to such high agility flight because the obstacle grid is constructed as a surface, abandoning useful information that could be used to build a volumetric obstacle field. For example, if the LIDAR detected an electric wire, it would be represented as a wall having the height of the measured wire. Fast sampling of point cloud data from the sensor can be used to build a 3D occupancy map of obstacles.

7.3.6 Collaborative obstacle field navigation

The current implementation of the obstacle avoidance framework has a potential functionality that can be applied to the collaborative autonomy of multiple UAVs. The target position of the optimization can be selected as an arbitrary position in the inertial frame. This could be a waypoint or even the position of other vehicles. A coarse path determined by global search or the cuboid obstacle field database might be shared by the ground control station or other vehicles moving toward the same destination point in the obstacle field. Therefore, the vehicles can share rough information about an unexplored region and can determine an initial safe route toward the destination a priori. For such autonomy of obstacle field navigation, the inter-vehicle communication and data sharing architecture are prerequisites.

APPENDIX A

ESTIMATION OF REQUIRED POWER

The required power can be estimated by a combination of momentum theory and blade element theory. Kong et al.[75] suggested a concise method to estimate it and this chapter is mainly referenced from their work.

Base starting point is the estimation of the induced velocity, v_i, at hovering condition

$$v_h = v_i = \sqrt{\frac{T}{2\rho A}} = \sqrt{\frac{W}{2\rho A}} \tag{95}$$

where T is the thrust, W is the weight, ρ is the air density, and A is the main rotor area, then the induced power at hovering is given by:

$$P_h = W v_h = W v_i = \frac{W^{\frac{3}{2}}}{\sqrt{2\rho A}} \tag{96}$$

In forward flight, the power required P_r can be approximated by the summation of the power components:

$$P_r = P_i + P_0 + P_p + P_t + P_{c/d} \tag{97}$$

where P_i is the induced power, P_0 is the profile power, P_p is the parasitic power to overcome the drag, P_t is the tail rotor power, and P_c/d is the power for climbing or descending.

Forward flight induced power can be calculated by the forward flight induced velocity

$$v_i = v_h \sqrt{\sqrt{\frac{1}{4}\left(\frac{V_\infty \sin\alpha}{v_h}\right)^4} - \frac{1}{2}\left(\frac{V_\infty \cos\alpha}{v_h}\right)^2} \tag{98}$$

where V_∞ is the forward speed and α is the angle of attack of the main rotor. The thrust T can be obtained by the acceleration

$$|T| = m|\vec{a} + (-\vec{g})| \tag{99}$$

so the induced power can be obtained Equations (98) and (100)

$$P_i = |T| v_i \tag{100}$$

The profile power can be estimated from the blade element theory

$$P_0 = \frac{\sigma C_{d_0}}{8}(1 + K\mu^2)P_h \tag{101}$$

The computation of $P_{c/d}$ is more complex. For instance, descending at slow forward speed is more expensive than hovering as long as $-2v_h \leq v_c \leq 0$. For Yamaha Rmax, the maximum descending velocity is chosen to be about $7ft/s$ at low forward speed, which is located in the region between hover and windmill state. And the threshold when the helicopter starts behaving like an airplane is chosen to be $15ft/s$. If the forward speed is lower than this threshold speed, the power can be approximated by

$$P_{c/d} = P_h\left(\kappa - \frac{v_c}{v_h}\right) \tag{102}$$

where κ is the induced power factor which is 1.0 for ideal case. As for climbing, the power under slower forward speed is:

$$P_{c/d} = P_h\left(\frac{v_c}{2v_h} + \sqrt{\left(\frac{v_c}{2v_h}\right)^2 + 1}\right) \tag{103}$$

For high forward speed, the climb power is equal to the rate of increase of potential energy and descending power is negative to reflect the fact that the helicopter can convert potential energy into kinetic energy.

The tail rotor power P_t typically varies between 3 to 5 percent of the main rotor power in normal flight.

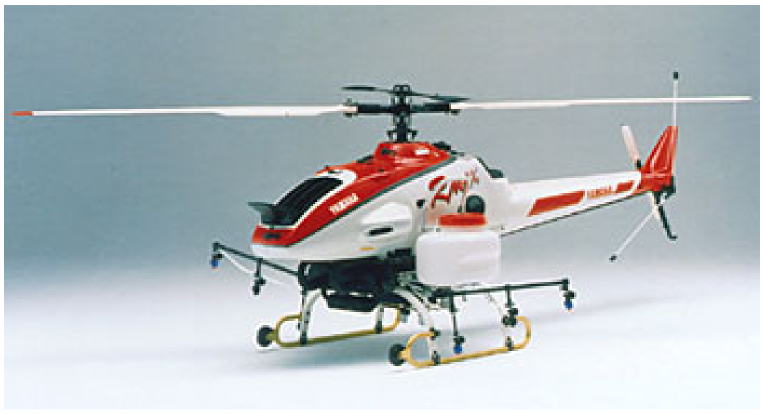


Figure 96: Original Yamaha RMAX is designed for agricultural and industrial use.

Table 6: Yamaha Rmax Specifications

main rotor diameter	3,115 mm
tail rotor diameter	545 mm
length	3.63 m (with main rotor)
width	2.0 m
height	1.22 m
weight	approx. 95 kg (209.4 lb)
engine	liquid-cooled 2-stroke cylinder, 21 hp

[*] http://www.yamaha-motor.co.jp/global/news/2002/02/06/sky.html

REFERENCES

[1] "Unmmaned aircraft systems roadmap 2005-2030," 2005.

[2] *IBM ILOG CPLEX*, 2010.

[3] Andert, F., "Online world modeling and path planning for an unmanned helicopter," *Autonomous Robots*, Vol. 27, No. 3, pp. 147–164, 2009.

[4] Bacerra, V. M., "Solving complex optimal control problems at no cost with psopt," *IEEE Multi-conference on Systems and Control*, pp. 1391–1396, 2010.

[5] Behnke, S., "Local multiresolution path planning," *RoboCup 2003: Robot Soccer World Cup VII*, Vol. 3020, pp. 332–343, 2004.

[6] Bellingham, J., Richards, A., and How, J., "Receding horizon control of autonomous aerial vehicles," *Preceedings of the American Control Conference*, Vol. 5, pp. 3741–3746, May 8-10 2002.

[7] Bellman, R., *Dynamic programming.* Princeton University Press, 1957.

[8] Benson, D. A., Huntington, G. T., Thorvaldsen, T. P., and Rao, A. V., "Direct trajectory optimization and costate estimation via an orthogonal collocation method," *Journal of Guidance, Control and Dynamics*, Vol. 29, No. 6, pp. 1435–1440, 2006.

[9] Betts, J. T., "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, pp. 193–207, 1998.

[10] Bone, E. and Bolkcom, C., "Unmanned aerial vehicles: Background and issues for congress," CRS RL31872, Congressional Research Service, April 25 2003.

[11] Boskovic, J. D., "Multi-layer control architecture for unmanned aerial vehicles," *Proceedings of the American Control Conference*, Vol. 3, pp. 1825–1830, 2002.

[12] Boukhtouta, A., Bedrouni, A., Berger, J., Bouak, F., and Guitouni, A., "A survey of military planning systems," *The 9th ICCRTS*, 2004.

[13] Broughton, R. N., "B-1 flight test progress report," *AIAA Aircraft Systems and Technology Conference*, AIAA, Aug. 21-23 1978.

[14] Bryson, A. E., *Dynamic optimization.* Addison Wesley Longman, 1999.

[15] Bryson, A. E. and Ho, Y. C., *Applied optimal control : optimization, estimation, and control.* Hemisphere Pub. Corp., 1975.

[16] Canny, J., *The complexity of robot motion planning.* The MIT Press, 1988.

[17] Canuto, C., *Spectral methods in fluid dynamics.* Springer series in computational physics, Springer-Verlag, 1988.

[18] Charlet, B., Levine, J., and Marino, R., "On dynamic feedback linearization," *Systems & Control Letters*, Vol. 13, No. 2, pp. 143–151, 1989.

[19] Cheng, V. H. L. and Sridhar, B., "Considerations for automated nap-of-the-earth rotorcraft flight," *Journal of the American Helicopter Society*, Vol. 36, No. 2, pp. 61–69, 1991.

[20] Cheng, V. H. L. and Sridhar, B., "Technologies for automating rotorcraft nap-of-the-earth flight," *Journal of the American Helicopter Society*, Vol. 38, No. 2, pp. 78–87, 1993.

[21] Clement, W., Gorder, P., Jewell, W., and Coppenbarger, R., "Real-time piloted simulation of fully automatic guidance and control for rotorcraft nap-of-the-earth (noe) flight following planned profiles," *AIAA Guidance, Navigation, and Control Conference*, Aug. 20-22 1990.

[22] Clough, B. T., "Unmanned aerial vehicles: Autonomous control challenges, a researcher's perspective," *Journal of Aerospace Computing, Information and Communication*, Vol. 2, No. 8, pp. 327–347, 2005.

[23] Connolly, C. I., Burns, J. B., and Weiss, R., "Path planning using laplace's equation," *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 2102–2106, 1990.

[24] Conway, B., "A survey of methods available for the numerical optimization of continuous dynamic systems," *Journal of Optimization Theory and Applications*, Vol. 152, No. 2, pp. 271–306, 2012.

[25] Coppenbarger, R. and Cheng, V., "Status of automated nap-of-the-earth rotorcraft guidance," *AIAA Guidance, Navigation and Control Conference*, Aug. 12-14 1991.

[26] Cox, T. H., Nagy, C. J., Skoog, M. A., and Somers, I. A., "Civil UAV capability assessment," tech. rep., NASA, 2004.

[27] Dadkhah, N. and Mettler, B., "Sensory predictive guidance in partially known environment," *AIAA Guidance, Navigation, and Control Conference*, Aug. 08-11 2011.

[28] DeBoor, C., *A Practical Guide to Splines.* Springer, 2001.

[29] DeBusk, W. M., "Unmanned aerial vehicle systems for disaster relief," *AIAA Infotech@Aerospace*, April 20-22 2010.

[30] Dickmanns, E. and Well, K., "Approximate solution of optimal control problems using third order hermite polynomial functions," *Lecture Nonte in Computer Science*, Vol. 27, pp. 158–166, Springer Berlin / Heidelberg, 1975.

[31] Dijkstra, E. W., "A note on two problems in connexion with graphs," *Numerische Mathematik*, Vol. 1, No. 1, pp. 269–271, 1959.

[32] Dolinskaya, I. S. and Smith, R. L., "Obstacle-avoiding fastest paths in anisotropic media," tech. rep., University of Michigan, Nov. 6 2008.

[33] Drozeski, G. R., Saha, B., and Vachtsevanos, G. J., "A fault detection and reconfigurable control architecture for unmanned aerial vehicles," *Proceedings of the IEEE Aerospace Conference*, pp. 1–9, 2005.

[34] Drud, A., *CONOPT Manual.* ARKI Consulting and Development, Bagsvaerd, Denmark.

[35] Fahroo, F. and Ross, I. M., "Second look at approximating differential inclusions," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 1, pp. 131–133, 2001.

[36] Ferguson, D., Likhachev, M., and Stentz, A., "A guide to heuristic-based path planning," *International Conference on Automated Planning and Scheduling*, pp. 9–18, 2005.

[37] Fliess, M., Levine, J., Martin, P., and Rouchon, P., "Flatness and defect of non-linear systems: introductory theory and examples," *International Journal of Control*, Vol. 61, No. 6, pp. 1327–1361, 1995.

[38] Frazzoli, E., *Robust hybrid control for autonomous vehicle motion planning.* PhD thesis, Massachusetts Institute of Technology, 2001.

[39] Frazzoli, E., Dahleh, M. A., and Feron, E., "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance Control and Dynamics*, Vol. 25, No. 1, pp. 116–129, 2002.

[40] Funk, J. E., "Terrain following control based on an optimized spline model of aircraft motion," Tech. Rep. ASD/XR-TR-75-22, Aeronutical System Division Wright-Patterson AFB, 1975.

[41] Funk, J. E., "Optimal-path precision terrain following system," *Journal of Aircraft*, Vol. 14, No. 2, pp. 128–134, 1976.

[42] Garg, D., Patterson, M., Hager, W. W., Rao, A. V., Benson, D. A., and Huntington, G. T., "A unified framework for the numerical solution of optimal control problems using pseudospectral methods," *Automatica*, Vol. 46, No. 11, pp. 1843–1851, 2010.

[43] Gasparetto, A. and Zanotto, V., "A new method for smooth trajectory planning of robot manipulators," *Mechanism and Machine Theory*, Vol. 42, No. 4, pp. 455–471, 2007.

[44] Geyer, M. S. and Johnson, E. N., "3D obstacle avoidance in adversarial environments for unmanned aerial vehicles," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Aug. 21-24 2006.

[45] Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H., "User's guide for NPSOL 5.0," Technical Report SOL 86-1, 1998.

[46] Gill, P. E., Murray, W., and Saunders, M. A., "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM REVIEW*, Vol. 47, No. 1, pp. 99–131, 2005.

[47] Goerzen, C., Kong, Z., and Mettler, B., "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *Journal of intelligent & robotic systems*, Vol. 57, No. 1, pp. 65–100, 2010.

[48] Graham-Rowe, D., "Cheap drones could replace search-and-rescue helicopters," *The New Scientist*, Vol. 207, No. 2769, pp. 20–20, 2010.

[49] Gupta, G. S., Demidenko, S., and Messom, C., "Obstacle avoidance in a collaborative environment," *I2MTC 2008 - IEEE International Instrumentation and Measurement Technology Conference*, Victoria, Vancouver Island, Canada, May 12-15 2008.

[50] Halpern, M. E., "Optimal trajectories for aircraft terrain following and terrain avoidance: A literature review update," Tech. Rep. ARL-TR-5, Defence Science adn Technology Organization Aeronautical Research Laboratory, March 1993.

[51] Hao, Y. and Agrawal, S. K., "Planning and control of ugv formations in a dynamic environment: A practical framework with experiments," *Robotics and Autonomous systems*, Vol. 51, No. 2, pp. 101–110, 2005.

[52] Hargraves, C. R. and Paris, S. W., "Direct trajectory optimization using nonlinear programming and collocation," *Journal of Guidance*, Vol. 10, pp. 338–342, 1987.

[53] Hart, P. E., Nilsson, N. J., and Raphael, B., "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, Vol. 4, No. 2, pp. 100–107, 1968.

[54] Hewlett, J. K., Schulein, G., and Mansur, M. H. "practical approach to obstacle field route planning for unmanned rotorcraft," *American Helicopter Society 60th Annual Forum*, June 7 2004.

[55] Huade, L. and Suraweera, F., "Introduction and solutions to noe path planning with incomplete information," *Proceedings of the IEEE Region 10 Conference on Computer, Communication, Control and Power Engineering*, Vol. 4, pp. 518–521, 1993.

[56] Hwang, Y. K., "Gross motion planning - a survey," *ACM Computing Surveys*, Vol. 24, No. 3, pp. 219–291, 1992.

[57] Hwangbo, M., Kuffner, J., and Kanade, T., "Efficient two-phase 3D motion planning for small fixed-wing UAVs," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1035–1041, 2007.

[58] Inanc, T., Misovec, K., and Murray, R. M., "Nonlinear trajectory generation for unmanned air vehicles with multiple radars," in *Proceedings of the 43rd IEEE Conference on Decision and Control*, Vol. 4, pp. 3817–3822, Dec. 2004.

[59] Jacobson, D. H. and Mayne, D. Q., *Differential dynamic programming.* American Elsevier Pub. Co., 1970.

[60] Jadbabaie, A., *Receding Horizong Control of Nonlinear Systems: A Control Lyapunov Approach.* PhD thesis, California Institute of Technology, 2000.

[61] Johnson, D. S. and Garey, M. R., *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman & Co, San Francisco, 1979.

[62] Johnson, E. N. and Kannan, S. K., "Adaptive trajectory control for autonomous helicopters," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 3, pp. 524–538, 2005.

[63] Johnson, E. N., Mooney, J. G., Ong, C., Sahasrabudhe, V., and Hartman, J., "Flight testing of nap-of-the-eath unmanned helicopter systems," *American Helicopter Society 67th Annual Forum*, May 3-5 2011.

[64] Johnson, E. N. and Schrage, D. P., "The georgia tech unmanned aerial research vehicle: GTMax," *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2003.

[65] Johnson, E. N., Schrage, D. P., Prasad, J. V. R., and Vachtsevanos, G., "UAV flight test programs at georgia tech," *AIAA Unmanned Unlimited Technical Conference, Workshop and Exhibit*, 2004.

[66] Johnson, K. L. and Wendl, M., "F15E terrain following system development," *AIAA Aerospace Design Conference*, Feb. 3-6 1992.

[67] Kailath, T., *Linear Systems.* Prentice-Hall, 1980.

[68] Kang, K., Prasad, J. V. R., and Dhingra, M., "Simulation evaluation of an adaptive optimal multiple obstacle avoidance algorithm for autonomous UAVs," *American Helicopter Society 67th Annual Forum*, May 3-5 2011.

[69] Kavraki, L. E., Kolountzakis, M. N., and Latombe, J. C., "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 1, pp. 166–171, 1998.

[70] Kawajir, Y., Laird, C., and Wachter, A., *Introduction to Ipopt: A tutorial for downloading, installing, and using Ipopt*, Rev. 1289, Aug. 26 2008.

[71] Khatib, O., "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, Vol. 5, No. 1, pp. 90–98, 1986.

[72] Kilmer, F. G., Kilmer, R. L., Swenson, H. N., and Woodward, A. C., "A helicopter terrain following system for terminal area operations," *AIAA/AHS/ASEE Aircraft Design Systems and Operations Meeting*, Oct. 14-16 1985.

[73] Kim, E., *Optimal helicopter trajectory planning for terrain-following flight.* PhD thesis, Georgia Institute of Technology, 1990.

[74] Kim, H. J., "Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles," *Proceedings of the American Control Conference*, Vol. 5, pp. 3576–3581, 2002.

[75] Kong, Z., Korukanti, V. R., and Mettler, B., "Mapping 3D guidance performance using approximate optimal cost-to-go function," *Proceedings of AIAA Guidance, Navigation, and Control Conference*, Aug. 10-13 2009.

[76] Koren, Y. and Borenstein, J., "Potential field methods and their inherent limitations for mobile robot navigation," *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1398–1404, 1991.

[77] Krachmalnick, F. M., Vetsch, G. J., and Wendl, M. J., "Automatic flight control system for automatic terrain-following," *Journal of Aircraft*, Vol. 5, pp. 168–175, 1968.

[78] Kuwata, Y. and How, J. P., "Three dimensional receding horizon control for UAVs," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Aug. 26-29 2004.

[79] Kuwata, Y., Karaman, S., Teo, J., Frazzoli, E., How, J. P., and Fiore, G., "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, Vol. 17, No. 5, pp. 1105–1118, 2009.

[80] Lantoine, G. and Russell, R. P., "A hybrid differential dynamic programming algorithm for robust low-thrust optimization," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Aug. 18-21 2008.

[81] Latombe, J. C., *Robot motion planning.* Kluwer Academic Publishers, 1991.

[82] LaValle, S. M., "Rapidly-exploring random trees: A new tool for path planning," *Computer and Information Science(Mendeley)*, Vol. 129, No. 98-11, pp. 98–11, 1998.

[83] LaValle, S. M., *Planning algorithms.* Cambridge University Press, 2006.

[84] Lin, T. C. and Arora, J. S., "Differential dynamic programming technique for constrained optimal control," *Computational Mechanics*, Vol. 9, No. 1, pp. 27–40, 1991.

[85] Liu, L., "Voronoi diagram and gis-based 3D path planning," *17th International Conference on Geoinformatics*, Aug. 12-14 2009.

[86] Locke, W. M., "Cruise missile system design," *AIAA 1981 Annual Meeting and Technical Display*, May 12-14 1981.

[87] Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. M., "Constrained model predictive control: Stability and optimality," *Automatica*, Vol. 36, No. 6, pp. 789–814, 2000.

[88] Menon, P. K. A., Cheng, V. H. L., and Kim, E., "Optimal trajectory synthesis for terrain-following flight," *Journal of Guidance*, Vol. 14, pp. 807–813, 1991.

[89] Mettler, B., Kong, Z., Goerzen, C. and Whalley, M, "Benchmarking of obstacle field navigation algorithms for autonomous helicopters," *American Helicopter Society 66th Annual Forum*, May 11-13 2010

[90] Mettler, B. and Bachelder, E., "Combining on- and offline optimization techniques for efficient autonomous vehicle's trajectory planning," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Aug. 15-18 2005.

[91] Milam, M. B., *Real-time optimal trajectory generation for constrained dynamical systems.* PhD thesis, California Institute of Technology, 2003.

[92] Milam, M. B., Mushambi, K., and Murray, R. M., "A new computational approach to real-time trajectory generation for constrained mechanical systems," *Proceedings of the 39th IEEE Confernce on Decision and Control*, Vol. 1, pp. 845–851, Dec. 12-15 2000.

[93] Moon, J., *Mission-based guidance system design for autonomous UAVs.* PhD thesis, Georgia Institute of Technology, 2009.

[94] Moon, J. and Prasad, J. V. R., "Minimum-time approach to obstacle avoidance constrained by envelope protection for autonomous UAVs," *Mechatronics*, Vol. 21, No. 5, pp. 861–875, 2011.

[95] Muezzinoglu, M. K. and Inane, T., "Trajectory generation in guided spaces using ntg algorithm and artificial neural networks," *Proceedings of the 2006 American Control Conference*, pp. 5776–5781, June 14 -16 2006.

[96] Mujumdar, A. and Padhi, R., "Evolving philosophies on autonomous obstacle/collision avoidance of unmanned aerial vehicles," *Journal of Aerospace Computing, Information, and Communication*, Vol. 8, No. 2, pp. 17–41, 2011.

[97] Murtagh, B. A. and Sauders, M. A., *Minos 5.5 User's Manu.* System Optimization Laboratory , Department of Operations Research, Stanford University, Stanford, California.

[98] Nieuwstadt, J. M. *Trajectory generation for nonlinear control systems.* PhD thesis, California Institute of Technology, 1997.

[99] Oxley, P. C., "Terrain following and terrain avoidance algorithms," *Proceedings of the IEE Colloquium on Navigation, Guidance and Control on Aerospace*, pp. 2/1–2/3, 1989.

[100] Paris, S. W., Riehl, J. P., and Sjauw, W. K., "Enhanced procedures for direct trajectory optimization using nonlinear programming and implicit integration," *AIAA/AAS Astrodynamics Specialist Conference*, Vol. 2, pp. 852–870, Aug. 21-24 2006.

[101] Perez, T. L. and Wesley, M. A., "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, Vol. 22, No. 10, pp. 560–570, 1979.

[102] Platts, J. T., "Autonomy in unmanned air vehicles," *The Aeronautical Journal*, pp. 97–105, Feb. 2006.

[103] Pontryagin, L. S. and Neustadt, L. W., *The mathematical theory of optimal processes.* Gordon and Breach Science Publishers, 1986.

[104] Popken, D. A. and Cox, L. A., "Simulation-based planning for theater air warfare," *Proceedings of SPIE*, Vol. 5423, pp. 54–65, 2004.

[105] Prasad, J. V. R., Kang, K., Vu, A., Dhingra, M., and Johnson, E. N., "Optimal obstacle avoidance constrained by envelope protection for autonomous UAVs," *The 2010 Heli-Japan Conference*, Nov. 1-3 2010.

[106] Rao, A. V., Benson, D. A., Darbu, C., Patterson, M. A., Francolin, C., Sanders, I., and Huntington, G. T., "Algorithm 902: Gpops, a matlab software for solving multiple-phase optimal control problems using the gauss pseudospectral method," *ACM Transactions on Mathematical Software*, Vol. 37, No. 2, pp. 22–39, 2010.

[107] Redding, J., Amin, J. N., Boskovic, J. D., Kang, Y., Hedrick, K., Howlett, J., and Poll, S., "A real-time obstacle detection and reactive path planning system for autonomous small-scale helicopters," *AIAA Guidance, Navigation and Control Conference and Exhibit*, Aug. 20-23 2007.

[108] Riedel, F. W., "Design of a control system which minimizes the probability of clobber for a terrain following cruise missile," *AIAA Guidance and Control Conference*, Aug. 16-18 1976.

[109] Ro, K., Oh, J. S., and Dong, L., "Lessons learned: application of small UAV for urban highway traffic monitoring," *45th AIAA Aerospace Sciences Meeting and Exhibit*, Jan. 8-11 2007.

[110] Ross, I. M., *A Beginner's Guid to DIDO*. Elissar, LLC, 2001.

[111] Ross, I. M. and Fahroo, F., "Issues in the real-time computation of optimal control," *Mathematical and computer modelling*, Vol. 43, No. 9, pp. 1172–1188, 2006.

[112] Saunders, J. and Beard, R., "Reactive vision based obstacle avoidance with camera field of view constraints," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Aug.18 - 21 2008.

[113] Scherer, S., Singh, S., Chamberlain, L., and Saripalli, S., "Flying fast and low among obstacles," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2023–2029, 2007.

[114] Schouwenaars, T., *Safe Trajectory Planning of Autonomous Vehicles*. PhD thesis, Massachusetts Institute of Technology, 2005.

[115] Schwartz, A., *Theory and Implementation of Numerical Methods Based on Runge-Kutta Integration for Solving Optimal Control Problems*. PhD thesis, University of California Berkeley, 1996.

[116] Schwartz, J. T., "Survey of motion planning and related geometric algorithms," *Artificial Intelligence*, Vol. 37, No. 1-3, pp. 157–169, 1988.

[117] Semsch, E., Jakob, M., Pavlicek, D., and Pechoucek, M., "Autonomous UAV surveillance in complex urban environments," *Proceeding of SPIE*, Vol. 5423, pp. 82–85, 2004.

[118] Shim, D., Chung, H., Kim, H., and Sastry, S., "Autonomous exploration in unknown urban environments for unmanned aerial vehicles," *AIAA Guidance, Navigataion, and Control Conference*, 2005.

[119] Shimoda, S., Kuroda, Y., and Iagnemma, K., "Potential field navigation of high speed unmanned ground vehicles on uneven terrain," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2828–2833, 2005.

[120] Sholes, E., "Evolution of a UAV autonomy classification taxonomy," *Proceedings of the IEEE Aerospace Conference*, March 3-10 2007.

[121] Sobers, D. M., J., Chowdhary, G., and Johnson, E. N., "Indoor navigation for unmanned aerial vehicles," *AIAA Guidance, Navigation, and Control Conference*, 2009.

[122] Stentz, A., "Optimal and efficient path planning for partially known environments," *Intelligent Unmanned Ground Vehicles*, Vol. 388, pp. 203–220, 1997.

[123] Stentz, A., Kelly, A., Rander, P., Herman, H., Amidi, O., Mandelbaum, R., Salgian, G., and Pedersen, J., "Real-time, multi-perspective perception for unmanned ground vehicles," *Robotics Institute*, Paper 16, 2003.

[124] Stryk, O. V. and Bulirsch, R., "Direct and indirect methods for trajectory optimization," *Annals of Operations Research*, Vol. 37, pp. 357–373, 1992.

[125] Sugisaka, M., "Working robots for nuclear power plant desasters," *Proceedings of the 5th IEEE International Conference on Digital Ecosystems and Technologies Conference (DEST)*, pp. 358–361, May 31- June 3 2011.

[126] Sundar, S. and Shiller, Z., "Time-optimal obstacle avoidance," *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 3075–3080, 1995.

[127] Sundar, S. and Shiller, Z., "Optimal obstacle avoidance based on the hamilton-jacobi-bellman equation," *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 2, pp. 305–310, 1997.

[128] Takahashi, O. and Schilling, R., "Motion planning in a plane using generalized voronoi diagrams," *Robotics and Automation, IEEE Transactions on*, Vol. 5, No. 2, pp. 143–150, 1989.

[129] Twigg, S. S., Calise, A. J., and Johnson, E. N., "3D trajectory optimization for terrain following and terrain masking," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Aug. 21-24 2006.

[130] Vandapel, N., "Planning 3-D path networks in unstructured environments," *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 2005, pp. 4624–4629, 2005.

[131] Waller, M. C., Rigopoulos, J. G., Blackman, D. R., and Berreen, T. F., "Considerations in the application of dynamic programming to optimal aircraft trajectory generation," *Proceedings of the IEEE National Aerospace and Electronics Conference*, May 21-25 1990.

[132] Watanabe, Y.,Calise, A. J., and Johnson, E. N., "Vision-based obstacle avoidance for UAVs," *AIAA Guidance, Navigation, and Control Conference*, Aug. 20-23 2007.

[133] WHALLEY, M., SCHULEIN, G., and THEODORE, C., "Design and flight test results for a hemispherical ladar developed to support unmanned rotorcraft urban operations research," *American Helicopter Society 64th Annual Forum*, April 29-May 1 2008.

[134] WHITBECK, R. F. and WOLKOVITCH, J., "Optimal terrain-following feedback control for advanced cruise missiles [final report]," TR-1147-2, Office of Naval Research, 1982.

[135] WOEGINGER, G., "Exact algorithms for np-hard problems: A survey," *Lecture Note in Computer Science*, Vol. 2570, pp. 185–207, Springer, 2003.

[136] YOUNG, G. D. J., HARRINGTON, W. W., OVERDORF, R. L., and RACHOVITSKY, E., "Terrain following/terrain avoidance system concept development," *AIAA Guidance and Control Conference*, Aug. 9-11 1982.

VITA

Keeryun Kang started his career in aircraft system design and simulation after receiving the B.Sc. and M.Sc. degrees in Aerospace Engineering at Inha University, Korea, in 1995 and 1997, respectively. Since 1997, he has been in research position in Agency for Defense Development (ADD) as a senior research engineer. In ADD, he has worked for 6DOF simulation of aircraft system, guidance and control algorithm design and integration to the system, hardware-in-the-simulation of control algorithms, and the conceptual design and simulation of aircraft system.

He began his study for a Ph.D degree at Georgia Institute of Technology in fall 2007. In 2008 and 2009, he worked on the nonlinear adaptive control of helicopter slung-load control for accurate positioning on a moving target, and since 2009 he began to study on the real-time trajectory optimization for obstacle avoidance of rotary-wing UAVs. His research interests include adaptive control, trajectory optimization, optimal control, aircraft system dynamics, and real-time simulation of aerial vehicles.

Online Optimal Obstacle Avoidance for Rotary-wing Autonomous Unmanned Aerial Vehicles

Keeryun Kang

172 Pages

Directed by Dr. J.V.R. Prasad

This thesis presents an integrated framework for online obstacle avoidance of rotary-wing unmanned aerial vehicles (UAVs), which can provide UAVs an obstacle field navigation capability in a partially or completely unknown obstacle-rich environment. The framework is composed of a LIDAR interface, a local obstacle grid generation, a receding horizon (RH) trajectory optimizer, a global shortest path search algorithm, and a climb rate limit detection logic.

The key feature of the framework is the use of an optimization-based trajectory generation in which the obstacle avoidance problem is formulated as a nonlinear trajectory optimization problem with state and input constraints over the finite range of the sensor. This local trajectory optimization is combined with a global path search algorithm which provides a useful initial guess to the nonlinear optimization solver. Optimization is the natural process of finding the best trajectory that is dynamically feasible, safe within the vehicle's flight envelope, and collision-free at the same time. The optimal trajectory is continuously updated in real time by the numerical optimization solver, Nonlinear Trajectory Generation (NTG), which is a direct solver based on the spline approximation of trajectory for dynamically flat systems. In fact, the overall approach of this thesis to finding the optimal trajectory is similar to the model predictive control (MPC) or the receding horizon control (RHC), except that this thesis followed a two-layer design; thus, the optimal solution works as a guidance command to be followed by the controller of the vehicle.

The framework is implemented in a real-time simulation environment, the Georgia Tech UAV Simulation Tool (GUST), and integrated in the onboard software of

the rotary-wing UAV test-bed at Georgia Tech. Initially, the 2D vertical avoidance capability of real obstacles was tested in flight. Then the flight test evaluations were extended to the benchmark tests for 3D avoidance capability over the virtual obstacles, and finally it was demonstrated on real obstacles located at the McKenna MOUT site in Fort Benning, Georgia. Simulations and flight test evaluations demonstrate the feasibility of the developed framework for UAV applications involving low-altitude flight in an urban area.