Quad-Rotor Path Planning for Cluttered and Uncertain Environments

Yoshihide Arai

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Aeronautics & Astronautics

University of Washington

2021

Committee:

Behçet Açikmeşe

Mehran Mesbahi

Program Authorized to Offer Degree:
Aeronautics & Astronautics

University of Washington

Abstract

Quad-Rotor Path Planning for
Cluttered and Uncertain Environments

Yoshihide Arai

Chair of the Supervisory Committee:
Professor Behçet Açikmeşe
William E. Boeing Department of Aeronautics & Astronautics

This thesis investigates the validity of the path planning algorithm that was developed in a previous study for cluttered and uncertain environments with high-performance and low-performance computing environments. Using Successive Convexification (SCVX) and compound State-Triggered Constraints (STCs), the path planning of a small unmanned aerial system flying through obstacles is assessed. The obstacles are placed with uniform distribution along with the flight course. Various configurations of the obstacles are used in the path-planning computation and the distribution of each computation time and obstacle violations are discussed to assess the path-planning. In evaluating the performance of SCVX, various temporal nodes and maximum number of SCVX iterations are used in the simulation. To assess compound STCs, the dynamics of the vehicle are two quad-rotors connected with a beam-like bar. The simulations are run with both high performance (i.e. normal CPU of a laptop computer) and low performance (i.e. underclocked CPU imitating on-board processor). The results show that the path-planning method is effective and reliable for cluttered and uncertain environments since the computing times converge at certain times that are enough for real-time computation with high-performance computing environments (i.e. normal laptop CPU). In addition, a method that may be able to improve obstacle avoidance and performance with low-performance computing environments is proposed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

$\mathbb{R}$	The set of real numbers
$\mathbb{R}_+$	The set of non-negative real numbers
$\mathbb{R}_{++}$	The set of positive real numbers
$\mathbb{R}^n$	The space of n-dimensional vectors
$\mathbb{R}^{m\times n}$	The space of $m \times n$-dimensional matrices
$\hat{e}_j$	Unit vector pointing along j^{th}-axis
t_f	Time of flight
K	The number of temporal node
k	k^{th} temporal node
$g(\cdot)$	Trigger function $g : \mathbb{R}^{n_z} \to \mathbb{R}$
$c(\cdot)$	Constraint function $c : \mathbb{R}^{n_z} \to \mathbb{R}$
z	Solution variable of an optimization problem
n_g	The number of trigger conditions
n_c	The number of constraint conditions
α_j	Non-negative slack variable
r_i	Quad-rotor initial position vector
r_f	Quad-rotor final position vector
l_o	Payload length
R_o	Obstacle radius
$r_{o,l}$	Payload length
A, B, E	Matrices describing the 3-DoF quad-rotor dynamics

r	Quad-rotor position vector
v	Quad-rotor velocity vector
u	Thrust vector
m	quad-rotor mass
k_d	Drag coefficient
g	Gravitational acceleration
T_{min}	Minimum allowable thrust magnitude
T_{max}	Maximum allowable thrust magnitude
θ_{max}	Minimum allowable tilt angle
w_o	Minimum spacing enforced around quad-rotor
A_d, B_d, E_d	Matrices describing the 3-DoF quad-rotor discrete time dynamics
$\bar{\nu}$	Virtual control terms
z_k^*	Solution obtained during the previous iteration
$\epsilon_{tr}, \epsilon_{vc}$	Specified thresholds

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor, Professor Behçet Açikmeşe. I am so glad that he is my advisor and appreciate the research opportunity at his wonderful lab, ACL. Thanks to his intelligent and insightful advice, I could accomplish this research.

I would like to thank Professors Mehran Mesbahi for serving on my reading committee. I thank you for accepting my offer willingly and taking the time.

I would like to thank lab members at ACL. Especially, I would like to thank Michael Szmuk for providing me with the algorithm of the previous research and teaching about it; Kazuya Echigo for helping me to find the topic about my research and teaching me the ABC of research; Margaret Skye Mceowen for helping me to join ACL by sharing her research topic as my first research theme.

I would like to thank the University of Washington Aeronautics & Astronautics staff. In particular, I would like to thank Danyel Hacker, Academic Adviser for her kind and sound academic advice during the MS program and even before I enter UW.

I would like to thank officers at the Ground Staff Office of the Japan Ground Self-Defense Force. Especially, I would like to thank Lieutenant Colonel Junpei Yamashita for supervising this program; Lieutenant Colonel Takaaki Fuwa for supporting my life in the US.

I would like to thank my undergraduate advisor, Professor Masanori Harada at the National Defense Academy of Japan. If I had not taken his control engineering class and he had not been my advisor during my undergraduate, my academic career would not be what it is today. Also, I thank you for the advice for my MS research.

I would also like to thank classmates whom worked hard with together during this MS program. Collaborating on difficult assignments and spending the weekends together have

become unforgettable memories.

DEDICATION

to my dear family and friends

Chapter 1

INTRODUCTION

Over the past few years, UAVs (unmanned aerial vehicles) have become important and necessary for various fields such as transportation, agriculture, disaster relief, and military. During the flight mission, UAVs may have to avoid collisions from obstacles. This is more likely when UAVs are used in uncertain environments like disaster scenes or battlefields. Figure 1.1 shows an example of a drone in a disaster site. [1] Therefore the collision avoidance is crucial for UAVs that act in such environments. Also, during the flight operations in cluttered and uncertain environments, the real-time path planning is required to deal with the Obstacles which suddenly appears such as falling objects.

Figure 1.1: Drone that acts in cluttered environment

[1] https://www.offiziere.ch/?p=33199

In order to avoid collisions, the following procedure can be used. First, detecting the obstacles. It is obvious that detecting and recognizing the obstacles is necessary to avoid them. This procedure is done by sensors such as stereovision and radars. After the detection of the obstacles, computing the collision-free trajectories is the second step. This path planning process is becoming a more and more active area of research over the past few decades. It is not easy to compute collision-free optimal trajectory since the dynamics and constraints can be complicated to handle various applications. Thus, many methods have been introduced for this process. The last process is controlling the UAVs to follow the trajectories. Feedback control such as MPC (Model Predictive Control) is often used for this process. Therefore, collision avoidance is performed by the above procedure. Figure 1.2 illustrates the procedure for obstacle avoidance.

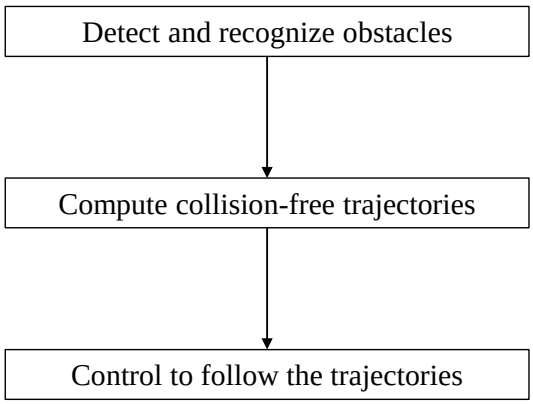


Figure 1.2: Procedure for obstacle avoidnance

In this thesis, only the second procedure, computing trajectories, is discussed. In terms of the path planning method, various kinds of algorithms have been developed. Among the optimization methods, convex optimization is used in various applications. The advantage of convex optimization is that it is fast and reliable since it has a property that every local optimum is a global optimum. However, some optimal control problems have non-convexities.

To solve non-convex optimal control problems, the Successive Convexification (SCVX) was introduced in [1]. Additionally, the research area of how to handle complicated constraints is also getting more and more active. In general, handling such complicated constraints needs a computationally expensive algorithm. Thus State-Triggered Constraints (STCs) are introduced in [2] to handle complicated constraints without relying on a computationally expensive algorithm. And the generalized form of STCs, compound STCs is introduced in [3]. Compound STCs are the algorithm that has the potential for various applications since they can handle complicated discrete decisions (e.g. a drone with complicated shape flying through obstacles).

In this thesis, Successive Convexification and compound State-Triggered Constraints are used to compute the collision-free optimal trajectories. An application of SCVX and compound STCs for quad-rotor path planning is introduced in [4]. This thesis uses that algorithm and investigates its validity for cluttered and uncertain environments assuming that a quad-rotor acts in such environments. The scenario that two identical quad-rotors connected with a beam-like payload fly through cluttered areas is used in the simulation. The simulation is run with various obstacle configurations to investigate the validity of the path planning algorithm for uncertain environments. Distributions of the computation time and violations of the obstacles are used for the assessment. In addition, the same simulation is run with a low-performance computing environment (i.e. underclocked CPU) to assess the reliability of the algorithm in such an environment. Compound STCs are used to handle discrete decisions due to the shape of the quad-rotors, and SCVX is used to solve the non-convex optimal control problem. The results show that the path-planning method is effective and reliable for cluttered and uncertain environments. The computing times converge at certain times that are enough for real-time computation with high-performance computing environments (i.e. normal laptop CPU). In addition, the method that may be able to improve obstacle avoidance and performance with low-performance computing environments is proposed.

1.1 Thesis Outline

This thesis is organized as follows. In Chapter 2, the theoretical background for this research is described. In Chapter 3, the scenario, formulations, and results of the path planning simulation is stated. In Chapter 4, the simulation results with the low-performance computing environment is presented. Lastly, in Chapter 5, concluding remarks are stated.

Chapter 2
THEORETICAL BACKGROUND

This chapter presents the theoretical background for this thesis. The theories used in this thesis are roughly divided into the following two: *Successive Convexification* and *compound State-Triggered Constraints*.

2.1 Successive Convexification

Successive Convexification (SCVX) is a method that solves non-convex continuous-time optimal control problems by solving a sequence of convex discrete-time optimization subproblems as convexification literally means conversion to convex problems. SCVX was first introduced in [1] to solve non-convex optimal control problems with nonlinear system dynamics. SCVX can solve problems that contain non-convex dynamics and non-convexities in state and control constraints. Each subproblem is an Second-Order Cone Programming (SOCP) and it is derived from the original problem by linearizing non-convexities about the previous iteration. Successive convexification roughly consists of *discretization* and *linearization*.

The discretization step discretizes the time horizon of the optimal control problem into $K-1$ temporal intervals. The length of the each time interval is $\Delta t := t_f/(K-1)$. For each node $k \in \mathcal{K} := \{1, 2, ..., K\}$, the time is given by $t_k := (k-1)\Delta t$.

The linearization step linearizes the non-convexities about the solution of the k^{th} iteration. This step generates a convex subproblem that is solved to full optimality, resulting in a new solution for the $(k+1)^{th}$ iteration. This successive process is repeated until the solution converges. However, the linearization step has two issues: *artificial infeasibility* and *artificial unboundedness*.

Artificial infeasibility can occur during the convergence process when the linearization is

unfavorable for feasibility. This issue occurs frequently during the first few SCVX iterations. To resolve this issue, *virtual control* terms are added to the discrete-time dynamics. Artificial unboundedness occurs when the linearization of an iterate results in constraints that accept an unbounded cost. To mitigate this issue, the cost with terms that work as soft *trust regions* defined around the previous iteration is added to the cost function.

More detailed formulations of SCVX for this research are discussed in Chapter 3.

2.2 Compound State-Triggered Constraints

Compound State-Triggered Constraints (STCs) are novel algorithm that can deal with discrete decisions without using discrete decision variables. Compound STCs were introduced in [3] as a generalization form of scalar STCs to solve powered-descend guidance rocket landing problems with discrete decisions. And compound STCs were applied to quad-rotor path planning in [4]. Not using discrete variables, compound STCs use continuous variables, which are compatible with continuous optimization such as successive convexification. Compound STCs enable us to deal with complex constraints (e.g. vehicles with complicated shapes flying through complex obstacles) with relatively low computation load compared to existing algorithms such as mixed-integer programming.

The formulation of scalar STCs is obtained by

$$g(z) < 0 \Rightarrow c(z) \leq 0, \tag{2.1}$$

where $z \in \mathbb{R}^{n_z}$ represents the solution variable of an optimization problem; $g(z) : \mathbb{R}^{n_z} \to \mathbb{R}$ is a trigger function and $c(z) : \mathbb{R}^{n_z} \to \mathbb{R}$ is a constraint function. The inequality $g(z) < 0$ is called a trigger condition and $c(z) \leq 0$ is called constraint condition. Hence, a continuous formulation that is equivalent to 2.1 is given by

$$h(z) := \hat{\sigma}(z) \cdot c(z) \leq 0, \tag{2.2}$$

where $\hat{\sigma}(z) := -\min(0, g(z))$.

The formulations of compound STCs, generalized form of scalar STCs, which were introduced in [3], are given by

$$\text{And-Trigger with Or-Constraint :} \qquad \bigwedge_{j=1}^{n_g}(g_j(z) < 0) \Rightarrow \bigvee_{j=1}^{n_c}(c_j(z) \leq 0), \tag{2.3}$$

$$\text{Or-Trigger with Or-Constraint :} \qquad \bigvee_{j=1}^{n_g}(g_j(z) < 0) \Rightarrow \bigvee_{j=1}^{n_c}(c_j(z) \leq 0), \tag{2.4}$$

where $g(\cdot) : \mathbb{R}^{n_z} \to \mathbb{R}^{n_g}$ and $c(\cdot) : \mathbb{R}^{n_z} \to \mathbb{R}^{n_c}$ are redefined as vector-valued trigger and constraint functions, and where $g_j(z)$ and $c_j(z)$ represent the j^{th} entries of the vectors $g(z)$ and $c(z)$, respectively. The continuous formulations corresponding to the compound STCs are given by

$$h_{\wedge\vee}(z) := \left[\prod_{j=1}^{n_g}\hat{\sigma}_j(z)\right] \cdot \left[\prod_{j=1}^{n_c}(c_j(z) + \alpha_j)\right] = 0, \tag{2.5}$$

$$h_{\vee\vee}(z) := \left[\sum_{j=1}^{n_g}\hat{\sigma}_j(z)\right] \cdot \left[\prod_{j=1}^{n_c}(c_j(z) + \alpha_j)\right] = 0, \tag{2.6}$$

where $\alpha_j \in \mathbb{R}_+$ are non-negative slack variables and $\hat{\sigma}_j(z)$ is j^{th} elements of a non-negative vector-valued $\hat{\sigma}(z)$.

Chapter 3

PATH PLANNING FOR COLLISION AVOIDANCE

This chapter presents the simulation of the collision-free path planning for this thesis.

3.1 Scenario

In this thesis, the scenario that two identical quad-rotors that are connected with a beam like bar of length l_o are flying through the obstacle area as in the previous research [4] is considered. Figure 3.1 illustrates this scenario. Within the fixed final time t_f, the vehicle starts from the initial position r_i and reaches the final position r_f flying through obstacles. The motions of the vehicles are restricted in the horizontal plane. The reason why two identical quad-rotors that are connected by a beam-like payload are considered is to make the constraints challenging and to use compound STCs. Note that, since the two quad-rotors are identical, the mass, the size, and the velocity and control boundary conditions are identical.

3.2 Configuration of Obstacles

In order to assess the path planning algorithm for cluttered and uncertain environments, the randomly distributed obstacles are assumed for this thesis. Along with the flight course, 10 cylindrical obstacles with radius R_o are placed. The obstacles are uniformly distributed in the region, $r_{o,l} \in [-2, 2] \times [2, 13]$ m. The area where the obstacles are distributed is shown in Figure 3.2. And 100 cases of different configurations are prepared for the simulation. Note that the obstacles are placed $[2, 13]$ m for the north position axis to exclude the impossible quad-rotor's maneuver which enforces the vehicle to avoid the obstacles immediately after and before it starts the flight and stops the flight.

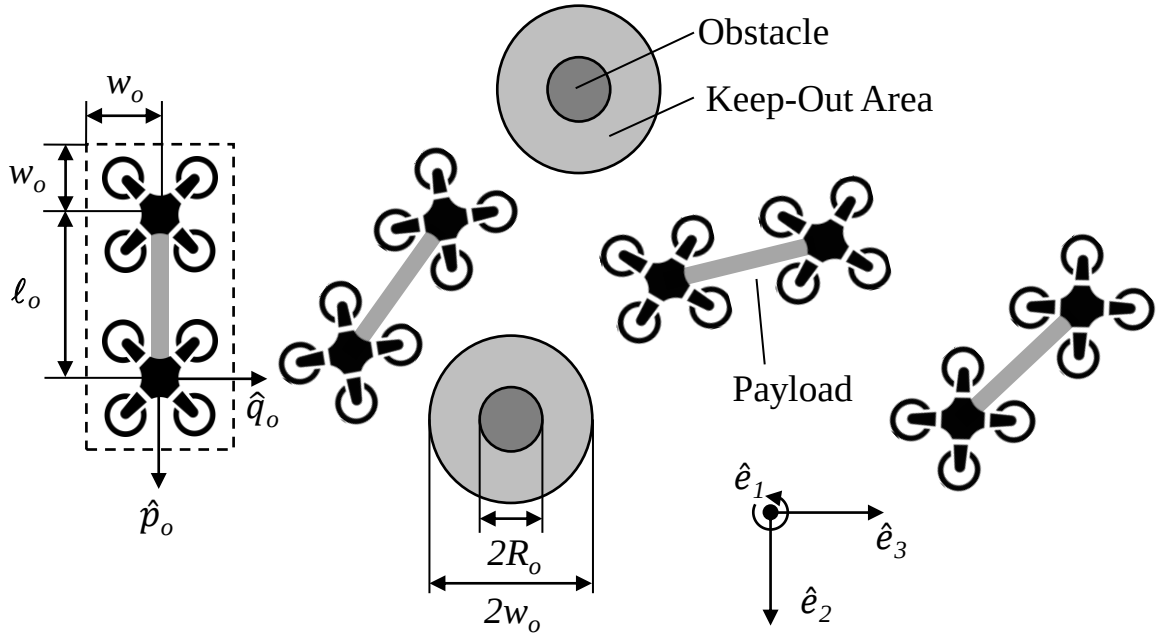


Figure 3.1: Scenario used in the simulation

Each obstacle is assumed infinitely long in the z-axis since avoiding objects by flying above or below is not desirable. If the objects are falling obstacles, avoiding horizontally is a suitable maneuver because it is not necessary to detect and assess the falling velocity.

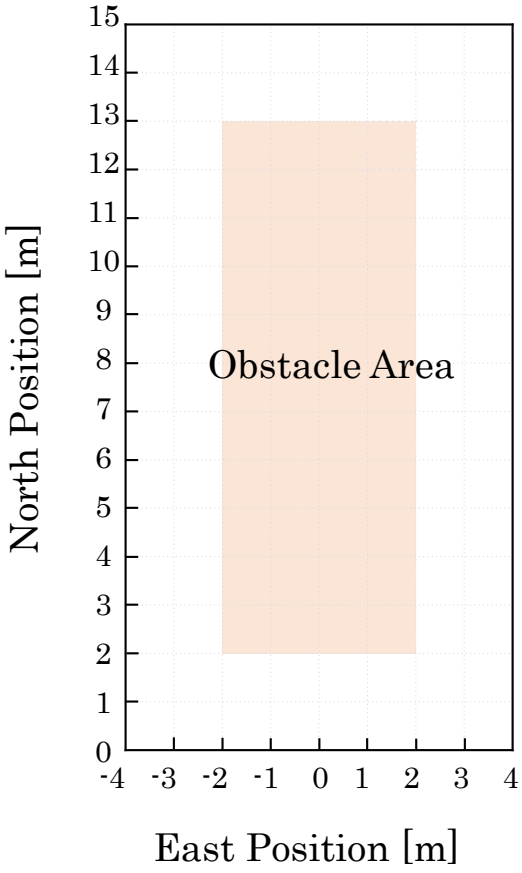


Figure 3.2: Area where the obstacles are placed

3.3 Quad-Rotor Dynamics

A 3-DoF simplified quad-rotor dynamics model which is used in the previous research[4] is assumed in this paper. The model is given by the following LTI model.

$$\dot{x}(t) = Ax(t) + Bu(t) + Ew,$$

$$A := \begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & -k_d I_{3\times3} \end{bmatrix},\ B := \frac{1}{m}\begin{bmatrix} 0_{3\times3} \\ I_{3\times3} \end{bmatrix}$$

$$x(t) := \begin{bmatrix} r^T(t) & v^T(t) \end{bmatrix}^T,\ E := -\hat{e}_4,\ w := g,$$

where $r(t) \in \mathbb{R}^3$ is the quad-rotor position vector, $v(t) \in \mathbb{R}^3$ is the quad-rotor velocity vector, $u(t) \in \mathcal{U} \subset \mathbb{R}^3$ is the thrust (control) vector, $m \in \mathbb{R}_{++}$ is the mass of the vehicle, $k_d \in \mathbb{R}_+$ is the drag coefficient, and $g \in \mathbb{R}_{++}$ is the local gravitational acceleration. Note that the drag model is simplified due to its linear dependence on $v(t)$.

$\mathcal{U}_1$, the first set of the control set $\mathcal{U}$ is the allowable thrust magnitudes of the quad-rotor. This is given by,

$$\mathcal{U}_1 := \{u \in \mathbb{R}^3 : 0 < T_{min} \leq \|u\|_2 \leq T_{max}\},$$

where T_{min} and T_{max} are the minimum and maximum allowable thrust magnitude respectively. Note that $\mathcal{U}_1$ is non-convex.

$\mathcal{U}_2$, the second set of the control set $\mathcal{U}$ is the allowable tilt angles of the quad-rotor. This is given by,

$$\mathcal{U}_2 := \{u \in \mathbb{R}^3 : \cos\theta_{max}\|u\|_2 \leq \hat{e}_1^T u\},$$

where $\theta_{max} \in (0°, 180°)$ is the maximum allowable tilt angle. Note that $\mathcal{U}_2$ is non-convex for $\theta_{max} > 90°$.

$\mathcal{U}_3$, the third set of the control set $\mathcal{U}$ is the thrust vectors with vertical control input equal and opposite to the weight of the quad-rotor to maintain a constant altitude. This is given by,

$$\mathcal{U}_3 := \{u \in \mathbb{R}^3 : \hat{e}_1^T u = mg\},$$

This has a non-empty interior when $T_{min} \leq mg \leq T_{max}$.

In this thesis, two-dimensional application is considered (see Section 3.1 for the detail). It requires only horizontal motion. Thus $\mathcal{U} = \mathcal{U}_1 \cap \mathcal{U}_2 \cap \mathcal{U}_3$ is convex.

3.4 Problem Formulation

The problem formulation which is used in the previous research[4] is considered in this thesis.

The logical implementation of the compound STC for this scenario is given by,

$$\bigwedge_{j=1}^{2}(g_j(\cdot,\cdot,\cdot) < 0) \Rightarrow \bigvee_{j=1}^{2}(c_j(\cdot,\cdot,\cdot) \leq 0), \tag{3.1}$$

$$g_1(r_1, r_2, r_{o,l}) := \hat{p}_o^T(r_1 - r_{o,l}) - w_o, \tag{3.2}$$

$$g_2(r_1, r_2, r_{o,l}) := \hat{p}_o^T(r_{o,l} - r_2) - w_o, \tag{3.3}$$

$$c_1(r_1, r_2, r_{o,l}) := \hat{q}_o^T(r_{o,l} - r_2) + w_o, \tag{3.4}$$

$$c_2(r_1, r_2, r_{o,l}) := \hat{q}_o^T(r_1 - r_{o,l}) + w_o, \tag{3.5}$$

where $\hat{p}_o := (r_2 - r_1)/\left\|r_2 - r_1\right\|_2$,$\hat{q}_o$ is orthogonal to $\hat{p}_o$ and $w_o \in \mathbb{R}_{++}$ is the minimum spacing enforced around each quad-rotor. Note that, the trigger conditions $\bigwedge_{j=1}^{2}(g_j(\cdot,\cdot,\cdot) < 0)$ means the condition in which the beam-like payload is likely to violate an obstacle as the obstacle is positioned between two quad-rotors. From (2.5) and (3.1) to (3.5), the following continuous formulation is obtained,

$$h(r_1, r_2, r_{o,l}) := \left[\prod_{j=1}^{2}\hat{\sigma}_j(r_1, r_2, r_{o,l})\right] \cdot \left[\prod_{j=1}^{2} c_j(r_1, r_2, r_{o,l} + \alpha_j)\right] = 0, \tag{3.6}$$

where $\hat{\sigma}_j(\cdot,\cdot,\cdot) := -\min(0, g_j(\cdot,\cdot,\cdot))$, and $\alpha_1, \alpha_2 \in \mathbb{R}_+$ are non-negative slack variables.

The fuel-optimal non-convex optimal control problem is shown in Problem A,

Problem A : *Fuel-Optimal Non-Convex Optimal Control Problem*

$$\underset{u}{\text{minimize}} \int_0^{t_f} \left(\|u_1(t)\|_2 + \|u_2(t)\|_2\right) dt$$

subject to :

$$\begin{gathered}
r_1(0) = r_{i,1},\ r_1(t_f) = r_{f,1},\ r_2(0) = r_{i,2},\ r_2(t_f) = r_{f,2}, \\
v_1(0) = v_2(0) = v_1(t_f) = v_2(t_f) = 0_{3\times 1}, \\
u_1(0) = u_2(0) = u_1(t_f) = u_2(t_f) = mg\hat{e}_1, \\
\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t) + \tilde{B}\tilde{u}(t) + \tilde{E}\tilde{w}, \\
u_1(t), u_2(t) \in \mathcal{U}_1 \cap \mathcal{U}_2 \cap \mathcal{U}_3, \\
h(r_1(t), r_2(t), r_{o,l}) = 0,\ \forall j \in \mathcal{N}_o, \\
\|v_1(t)\|_2 \leq v_{max},\ \|v_2(t)\|_2 \leq v_{max}, \\
\|r_1(t) - r_2(t)\|_2 = l_o.
\end{gathered}$$

where

$$\begin{gathered}
\tilde{A} := \mathbf{blkdiag}\{A, A\},\ \tilde{B} := \mathbf{blkdiag}\{B, B\},\ \tilde{E} := \mathbf{blkdiag}\{E, E\}, \\
\tilde{w} := [w\ \ w]^T,\ \tilde{x} := [x_1^T\ \ x_2^T]^T,\ \tilde{u} := [u_1^T\ \ u_2^T]^T.
\end{gathered}$$

Note that the control set is convex and constraints

$$\|r_1(t) - r_2(t)\|_2 = l_o \text{ and } h(r_1(t), r_2(t), r_{o,l}) = 0$$

are non-convex. Both of these non-convex constraints are handled by using SCVX.

The following Problem B shows the SOCP subproblem solved in the SCVX.

Problem B : ***SOCP Subproblem***

$$\underset{\bar{u},\bar{\nu}}{\text{minimize}}\, J(\bar{z}) + J_{tr}(\bar{z}) + J_{vc}(\bar{\nu})$$

subject to :

$$x_1 = x_{d,i},\ x_K = x_{d,f},\ u_1 = u_{d,i},\ u_K = u_{d,f},$$
$$x_k + 1 = A_d x_k + B_d^- u_k + B_d^+ u_{k+1} + E_d w_d + \nu_k,\ \forall k \in \bar{\mathcal{K}},$$
$$h(z_k^*) + \left.\frac{\delta h}{\delta z_k}\right|_{z_k^*} \delta z_k = 0,\ u_k \in \mathcal{U},\ \forall k \in \mathcal{K}.$$

where $J(\bar{z})$ is the oiginal objective, and $x_{d,i}$, $u_{d,i}$, $x_{d,f}$, and $u_{d,f}$ are the boundary conditions, and $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$, $A_d \in \mathbb{R}^{n_x \times n_x}$, $B_d^-, B_d^+ \in \mathbb{R}^{n_x \times n_u}$, $E_d \in \mathbb{R}^{n_x \times n_w}$, and $w_d \in \mathbb{R}^{n_w}$ are discrete dynamics, and $\bar{\mathcal{K}} := \mathcal{K}/K$, $\bar{u} := [u_1^T, \ldots, u_K^T]^T$, $z_k := [x_k^T, u_k^T, \alpha_k]^T$, $\bar{z} := [z_1^T, \ldots, z_K^T]^T$, and $\alpha_k \in \mathbb{R}_+^{n_\alpha}$ is a vector of non-negative slack variable, and $n_z = n_x + n_u + n_\alpha$. $J_{vc}(\bar{\nu})$ is a cost for virtual control terms $\bar{\nu} \in \mathbb{R}^{n_x}$ and $J_{tr}(\bar{z})$ is a cost for trust region and ϵ_{vc} , $\epsilon_{tr} \in \mathbb{R}_{++}$ are their specified thresholds.

The following Algorithm 1 shows the algorithm for SCVX.

Algorithm 1 Successive Convexification Algorithm

initialize z_k^*
discretize - compute $A_d, B_d^-, B_d^+, E_d, w_d$
for $i = 1$ **to** n_{SCVX_max} **do**
  linearize - compute $h(z_k^*)$ and $\delta h/\delta z_k|_{z_k^*}$
  solve SOCP
  if $J_{tr}(\bar{z}) < \epsilon_{tr}$ **and** $J_{vc}(\bar{\nu}) < \epsilon_{vc}$ **then**
    converged
  else if $i < n_{SCVX\_max}$ **then**
    $z_k^* \leftarrow z_k$
  end if
end for
return z_k

where n_{SCVX_max} is a maximum number of SCVX iterations. Within the specified maximum number of SCVX iterations, this algorithm successively linearizes and solves SOCP until the cost $J_{vc}(\cdot)$ and $J_{tr}(\cdot)$ are less than the specified thresholds.

3.5 Results

The results in this chapter are obtained by the following environments. The operating system of a laptop computer that runs the simulation is Windows 10 64-bit. It has a 1.8 GHz Intel Core i7-8550U CPU and 16GB of RAM. MATLAB is used to run the ECOS[5] solver through the CVX[6] parsing interface, and computation time is obtained by using `cvx_tic` and `cvx_toc` MATLAB function. During the simulation, the clock speed that the Windows Task Manager shows is approximately 3.0 GHz. Note that the clock speed is fluctuating a little while running the simulation.

The problem parameter is the following values:

$$r_{i,1} = (-0.25, 0)\,\mathrm{m}, \quad r_{i,2} = (0.25, 0)\,\mathrm{m},$$

$$t_f = 10\,\mathrm{s}, \quad r_{f,1} = (0, 14.9)\,\mathrm{m}, \quad r_{f,2} = (0.5, 14.9)\,\mathrm{m},$$

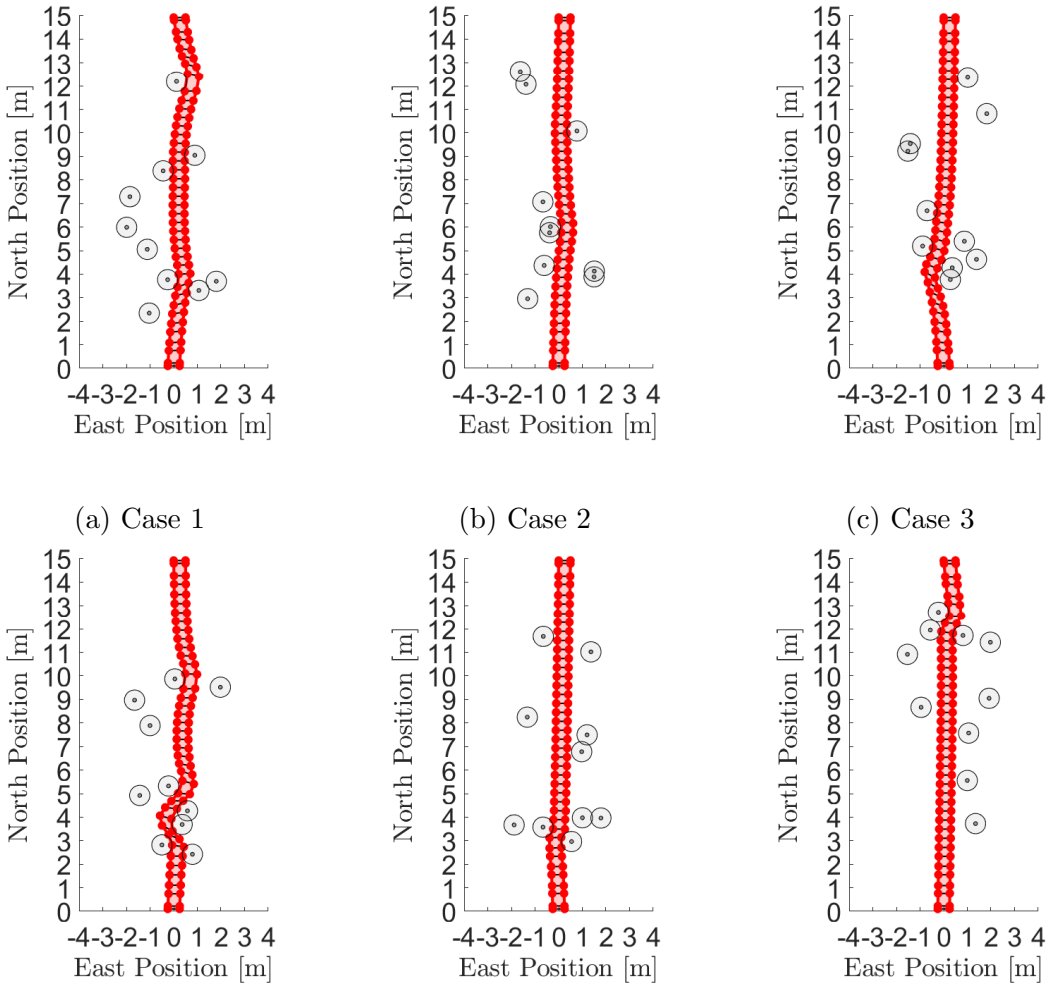
$$l_0 = 0.5\,\mathrm{m}, \quad m = 0.35\,\mathrm{kg}, \quad \theta_{max} = 45^\circ, \quad k_d = 0\,\mathrm{s}^{-1}$$

$$v_{max} = 3\,\mathrm{m/s}, \quad T_{min} = 2\,\mathrm{N}, \quad T_{max} = 5\,\mathrm{N}, \quad R_o = 0.08\,\mathrm{m},$$

$$w_o = 0.43\,\mathrm{m}, \quad g = 9.81\,\mathrm{m/s^2}, \quad W_{vc} = 10^5 \cdot I_{8\times 8},$$

$$W_{tr} = 50 \cdot \mathbf{blkdiag}\{I_{8\times 8}, 0_{4\times 4}\},$$

The temporal resolution of $K = 40$ and the maximum number of the SCVX iterations is 10 are used in the first experiment. The first 6 trajectories with different configurations of obstacles are shown in Figure 3.3. And Table 3.1 provides the statics of the computation time. Figure 3.5 illustrates the histogram of the computation time of the first experiment. 19 violations of the obstacles are observed and the number of the cases that fail to converge is 4. The 19 violations of the obstacles consist of 2 cases of converging fail and 17 cases of success in converging. Convergence failure means the algorithm can not return feasible solutions within the maximum SCVX iterations. The result shows that the distribution of computation time is reliable for the real-time path planning for uncertain and cluttered environments however the violation of the obstacles lacks reliability. By observing the trajectories that violate the obstacles, it can be presumed that the cause of the violation is the roughness of the discretization. The example case that violates the obstacles is shown in Figure 3.4. The violation happens at around 13 m in the north position. As the figure shows, the violation happens in the gap between each node. Hence, the different values of the temporal node, K are considered for next experiments. Also, in order to investigate the effect of maximum numbers of SCVX iterations, the different values of that are considered.



(a) Case 1 (b) Case 2 (c) Case 3

(d) Case 4 (e) Case 5 (f) Case 6

Figure 3.3: First 6 trajectories of the first experiment

Table 3.1: Statics of computation time for the first experiment[s]

Min	Max	Median	Mean	Std. Dev.
0.1363	0.6593	0.3450	0.3568	0.1162

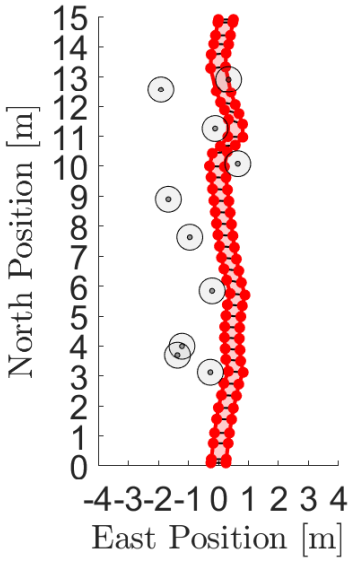


Figure 3.4: Example of the case which violates the obstacles

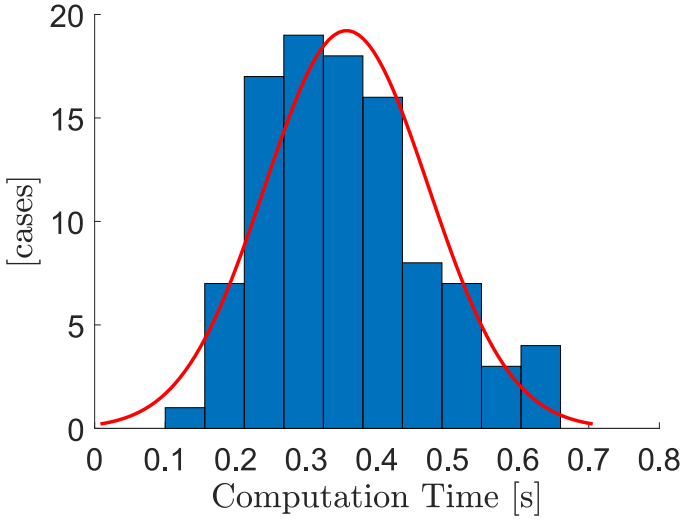


Figure 3.5: Histogram for the first experiment

In the next simulations, $K = 40, 50, 60$, and the maximum number of the iterations, 10,15, and 20 are used. Various combinations of the temporal nodes and SCVX iterations are simulated. Table 3.2, 3.3, and 3.4 show the statics of the computation time for $K = 40, 50, 60$ respectively. Figure 3.6 shows histograms for the computation time distributions of each case.

Table 3.2: Statics of computation time [s], $K = 40$

$K = 40$					
Max iteration	Min	Max	Median	Mean	Std. Dev.
=10	0.1363	0.6593	0.3450	0.3568	0.1162
Max iteration	Min	Max	Median	Mean	Std. Dev.
=15	0.0996	0.7351	0.2934	0.3112	0.1174
Max iteration	Min	Max	Median	Mean	Std. Dev.
=20	0.0988	0.9123	0.2911	0.3197	0.1409

Table 3.5, 3.6, and 3.7 shows the statics of convergence and obstacle violation for each combination. It is evident from these statics that significant improvement in the obstacle violation is observed when the temporal node is changed into 50 and 60. From the results, the following tendency is observed. The increase of the temporal node K leads to an increase in the computation time and convergence failure while the obstacle violation decreases. And the increase of the SCVX iteration leads to an increase in the standard deviation of the computation time distribution while the cases that fail to converge decrease.

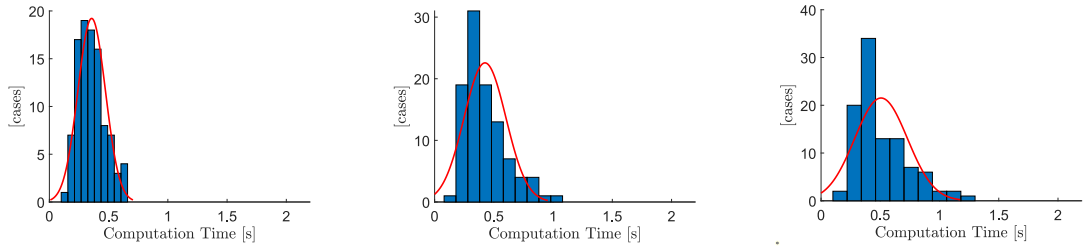
The statics of computation time and histograms when K=50,60 and the maximum number of the SCVX iteration is 10 show that the path-planning algorithm converges to certain times and it can be assumed to be reliable. The computation times converge around 0.4 s, and the standard deviations are around 0.2 s for both cases. It can be thought that this result is enough for real-time path planning. However, the 4 cases of obstacle violation can not

Table 3.3: Statics of computation time [s], $K = 50$

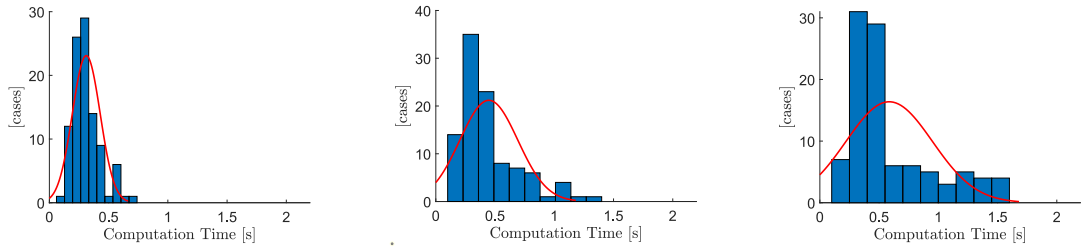
$K = 50$					
Max iteration	Min	Max	Median	Mean	Std. Dev.
=10	0.1479	1.0073	0.3759	0.4251	0.1766
Max iteration	Min	Max	Median	Mean	Std. Dev.
=15	0.1475	1.3161	0.3696	0.4475	0.2447
Max iteration	Min	Max	Median	Mean	Std. Dev.
=20	0.1457	1.6635	0.4029	0.5023	0.3320

Table 3.4: Statics of computation time [s], $K = 60$

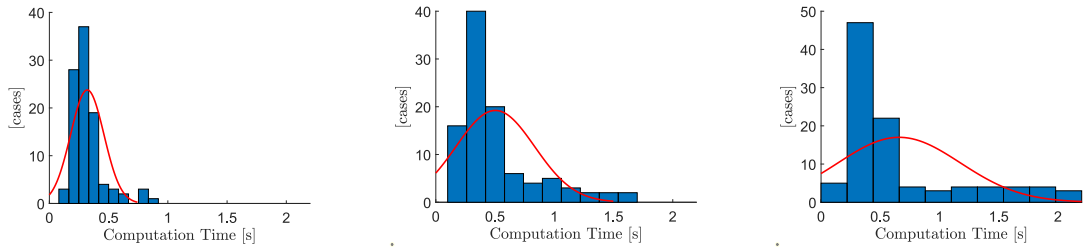
$K = 60$					
Max iteration	Min	Max	Median	Mean	Std. Dev.
=10	0.1624	1.2426	0.4372	0.5064	0.2224
Max iteration	Min	Max	Median	Mean	Std. Dev.
=15	0.1600	1.5726	0.4280	0.5853	0.3652
Max iteration	Min	Max	Median	Mean	Std. Dev.
=20	0.1628	2.1633	0.4264	0.6594	0.5164



(a) K=40, Max iteration=10 (b) K=50, Max iteration=10 (c) K=60, Max iteration=10



(d) K=40, Max iteration=15 (e) K=50, Max iteration=15 (f) K=60, Max iteration=15



(g) K=40, Max iteration=20 (h) K=50, Max iteration=20 (i) K=60, Max iteration=20

Figure 3.6: Histograms for the computation time distributions of each case

be overlooked. Observing the cases that fail to avoid the obstacles, it can be assumed that the dense obstacles are the cause of the obstacle violations. Figure 3.7 illustrates the case that violates the obstacles and shows that the violations occur in the area where the obstacles are dense. Dense obstacles can cause computational complexity and even make "dead end". Increasing the temporal nodes K resolves the issue of dense obstacles. However, the computation time will be longer and the "dead end" situation can not be resolved. SCVX has the disadvantage that it can not handle avoiding obstacles that make dead ends since it returns optimal solution for each discretized SOCP. To solve this problem, it is presumed that an idea like the one shown in Figure 3.8 will be needed. Approximating the dense obstacles can have good compatibility with SCVX.

Table 3.5: Statics of convergence and obstacle violation [cases], $K = 40$

$K = 40$		
Max iterations	Convergence failure	Obstacle violation
=10	5	19 (17 Converged, 2 Failed to converge)
Max iterations	Convergence failure	Obstacle violation
=15	4	19 (18 Converged, 1 Failed to converge)
Max iterations	Convergence failure	Obstacle violation
=20	4	19 (18 Converged, 1 Failed to converge)

Table 3.6: Statics of convergence and obstacle violation [cases], $K = 50$

$K = 50$		
Max iterations	Convergence failure	Obstacle violation
=10	20	4 (1 Converged, 3 Failed to converge)
Max iterations	Convergence failure	Obstacle violation
=15	13	4 (2 Converged, 2 Failed to converge)
Max iterations	Convergence failure	Obstacle violation
=20	8	4 (2 Converged, 2 Failed to converge)

Table 3.7: Statics of convergence and obstacle violation [cases], $K = 60$

$K = 60$		
Max iterations	Convergence failure	Obstacle violation
=10	22	4 (1 Converged, 3 Failed to converge)
Max iterations	Convergence failure	Obstacle violation
=15	20	4 (1 Converged, 3 Failed to converge)
Max iterations	Convergence failure	Obstacle violation
=20	14	4 (1 Converged, 3 Failed to converge)

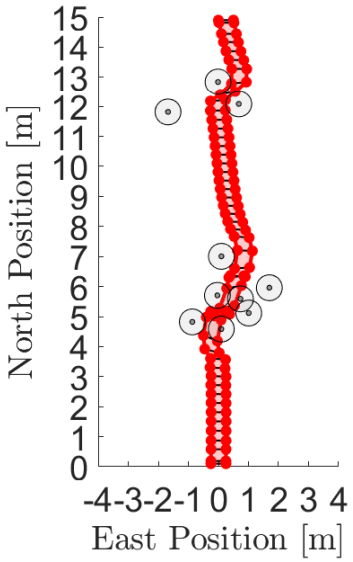


Figure 3.7: Example that fail to avoid the obstacles, $K = 50$

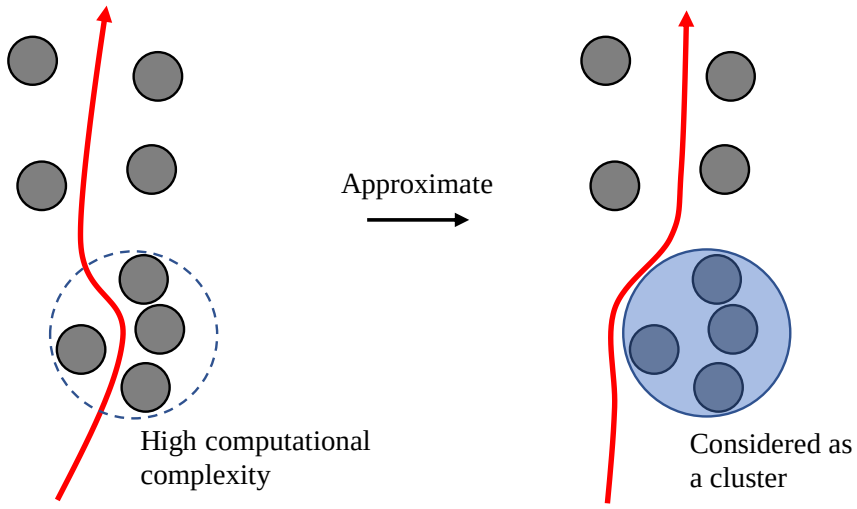


Figure 3.8: Proposal of method for setting constraints on obstacles

Chapter 4

ASSESSMENT FOR LOW PERFORMANCE ENVIRONMENT

The on-board processor of a quad-rotor, which is relatively lower performance than a usual computer is used to compute the trajectory for the actual application. Therefore, the path planning algorithm must be effective even with a low-performance processor for the actual implementation. In this chapter, the difference between a high-performance processor(i.e. computer) and a low-performance processor(i.e. on-board processor) is discussed.

4.1 Realization of Low Performance Environment

Low-performance environment is obtained by using the Windows setting to adjust the CPU clock speed shown in Figure 4.1.

In order to assess the relationship between decreasing the CPU clock speed and the performance of MATLAB, the benchmark test was done. Using the four different clock speeds, the relationship is discussed. MATLAB "`bench`" was used to run the benchmark test. MATLAB "`bench`" returns a 1-by-6 vector with the measured execution times. The six benchmarking tasks are listed in the following Table 4.1.

The following Table 4.2 illustrates the result of the benchmark test using the four different clock speeds.

From the result of the table above, it is observed that decreases in the clock speed correspond to the decreases in computational time. Therefore, using this realization of the low-performance environment is reasonable to simulate the path planning algorithm to assess the performance of the algorithm with the low-performance processor.

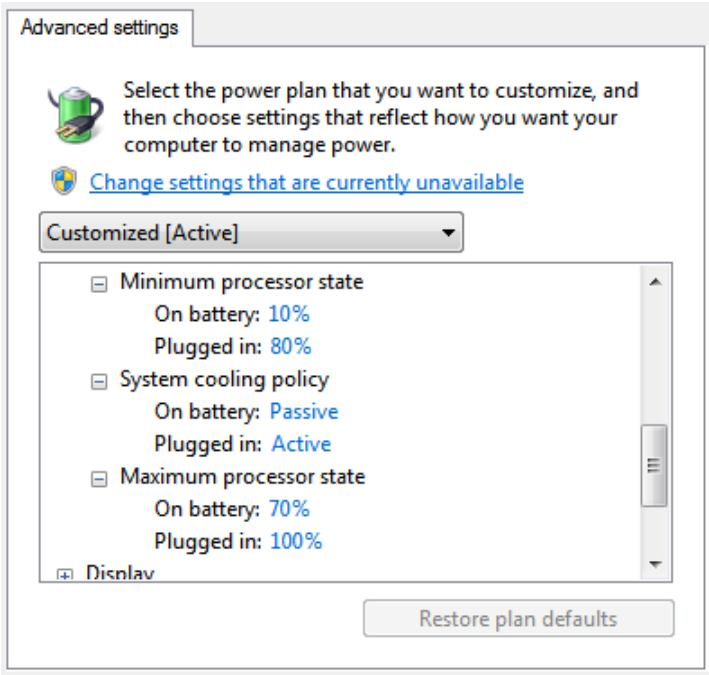


Figure 4.1: Screenshot of the setting

4.2 Results

The same environment as Chapter 3 except for the clock speed of the CPU is used in this simulation. Using the realization of the low-performance environment explained in Section 4.1, the clock speed is underclocked. The underclocked CPU clock speed observed in the Windows Task Manager during the simulation is around 1.0 GHz.

Table 4.3 and 4.4 show the statics of the computation time for $K = 50, 60$ respectively. Figure 4.2 shows histograms for the computation time distributions of each case. The simulation when $K = 40$ is omitted since the number of the cases that violate obstacles is 19 as shown in Chapter 3. The statics of convergence and obstacle violation are omitted as well since the results are the same as the simulation computed by the CPU with the normal clock speed.

Table 4.1: Benchmark tasks perfomed in MATLAB `bench`

Task	Description	Performance Factors
LU	Perform lu of a full matrix	Floating-point, regular memory access
FFT	Perform fft of a full vector	Floating-point, irregular memory access
ODE	Solve van der Pol equation with ode45	Data structures and MATLAB function files
Sparse	Solve a symmetric sparse linear system	Mixed integer and floating-point
2-D	Plot Lissajous curves	2-D line drawing graphics
3-D	Display colormapped peaks with clipping and transforms	3-D animated OpenGL graphics

Table 4.2: Benchmark result [s]

CPU clock speed	LU	FFT	ODE	Sparse	2-D	3-D	Sum
100%	0.8704	0.5786	0.5128	0.5875	0.674	0.6009	3.8242
75%	1.9408	0.7356	1.2926	1.1309	0.9325	0.9185	6.9509
50%	2.9067	1.0773	2.0687	1.545	1.299	0.9307	9.8274
25%	5.8934	2.0963	4.0485	3.0629	2.2684	2.1651	19.5346

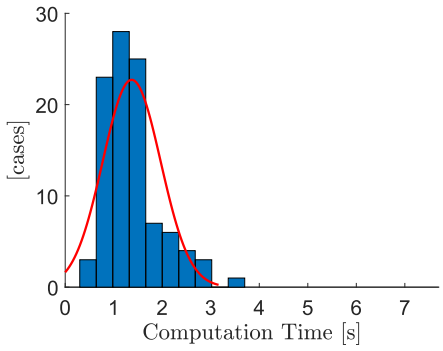
From the results, significant performance degradation is observed. The point to pay attention to is the standard deviations in this result. If the standard deviations of high-performance environments and low-performance environments are almost similar, it can be assumed that the path planning algorithm is even reliable in low-performance environments. However, the result in this experiment shows the computation distribution is much wider than one with high-performance environments. Wide computation time distribution is considered unreliable for the path planning for cluttered and uncertain environments. It can

Table 4.3: Statics of computation time [s], $K = 50$

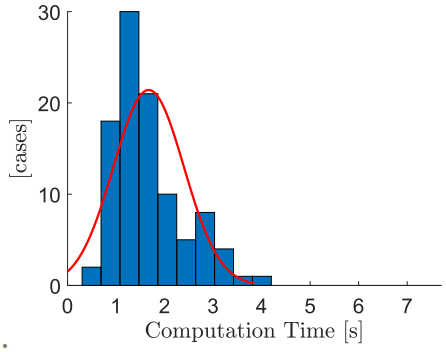
$K = 50$					
Max iteration	Min	Max	Median	Mean	Std. Dev.
=10	0.4725	3.6272	1.2339	1.3699	0.5969
Max iteration	Min	Max	Median	Mean	Std. Dev.
=15	0.4782	4.5053	1.2626	1.5443	0.8502
Max iteration	Min	Max	Median	Mean	Std. Dev.
=20	0.5061	5.9384	1.3174	1.7305	1.1569

Table 4.4: Statics of computation time [s], $K = 60$

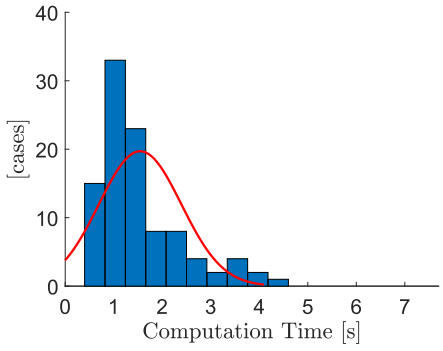
$K = 60$					
Max iteration	Min	Max	Median	Mean	Std. Dev.
=10	0.5328	4.1701	1.4707	1.6712	0.7263
Max iteration	Min	Max	Median	Mean	Std. Dev.
=15	0.5330	5.5405	1.4800	1.9836	1.2573
Max iteration	Min	Max	Median	Mean	Std. Dev.
=20	0.5404	7.6549	1.5026	2.2779	1.8049



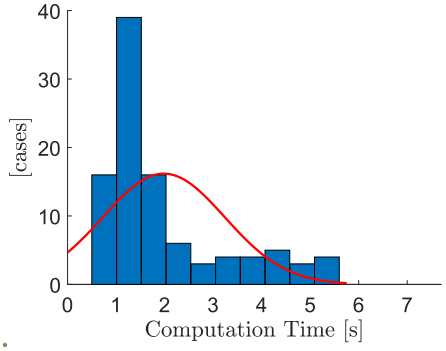
(a) K=50, Max iteration=10



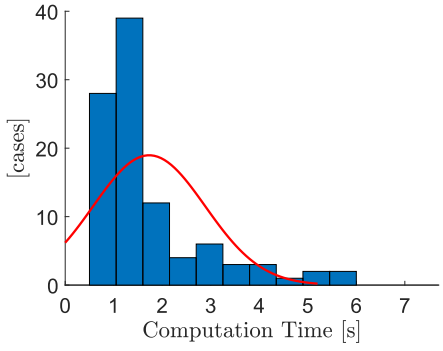
(b) K=60, Max iteration=10



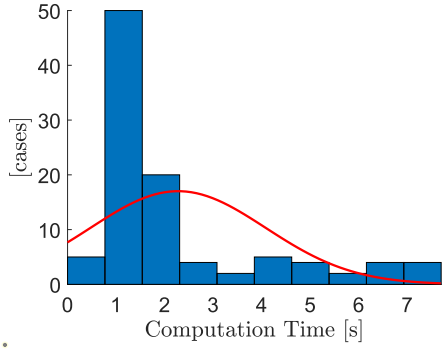
(c) K=50, Max iteration=15



(d) K=60, Max iteration=15



(e) K=50, Max iteration=20



(f) K=60, Max iteration=20

Figure 4.2: Histograms for the computation time distributions of each case with low-performance environments

be presumed that this significant performance degradation is due to the complexity of the obstacles as discussed in Chapter 3.

Chapter 5

CONCLUSION AND FUTURE WORK

This thesis investigates the validity of the path planning algorithm for cluttered and uncertain environments with high-performance and low-performance computing environments. The simulation results indicate that the algorithm is effective with high-performance environments since the distribution of computing time is reliable. However, there is room for improvement for the setting of the constraints and computation with low-performance environments. More effective constraints can be obtained by approximating the density obstacles as explained in Chapter 3. Using this approximation, it is presumed that the results with low-performance environments will be improved. Future work will be done with the following topics: using the more complicated shape of the vehicle to assess STCs, implementing the algorithm approximating the constraints for obstacles, and simulation with actual low-performance environments (i.e. on-board processor).

BIBLIOGRAPHY

[1] Yuanqi Mao, Michael Szmuk, and Behçet Açıkmeşe. Successive convexification of non-convex optimal control problems and its convergence properties. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 3636–3641. IEEE, 2016.

[2] Michael Szmuk, Taylor P Reynolds, and Behcet Acikmese. Successive convexification for real-time 6-dof powered descent guidance with state-triggered constraints. *arXiv preprint arXiv:1811.10803*, 2018.

[3] Michael Szmuk, Taylor Reynolds, Behcet Acikmese, Mehran Mesbahi, and John M Carson. Successive convexification for 6-dof powered descent guidance with compound state-triggered constraints. In *AIAA Scitech 2019 Forum*, page 0926, 2019.

[4] Michael Szmuk, Danylo Malyuta, Taylor P Reynolds, Margaret Skye Mceowen, and Behçet Açikmese. Real-time quad-rotor path planning using convex optimization and compound state-triggered constraints. *arXiv preprint arXiv:1902.09149*, 2019.

[5] Alexander Domahidi, Eric Chu, and Stephen Boyd. Ecos: An socp solver for embedded systems. In *2013 European Control Conference (ECC)*, pages 3071–3076. IEEE, 2013.

[6] Michael Grant and Stephen Boyd. Cvx: Matlab software for disciplined convex programming, version 2.1, 2014.

[7] Yuanqi Mao, Michael Szmuk, Xiangru Xu, and Behçet Açikmese. Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems. *arXiv preprint arXiv:1804.06539*, 2018.