SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY

Study of Efficient Mixed-Integer Planning for Autonomous Vehicles in Densely Cluttered Environments

by

Mosam Sanjay Dabhi

A technical report submitted in partial fulfillment for the Seminar presentation in the Electronics and Communication Engineering Department

October 2016

Declaration of Authorship

I, Mosam Sanjay Dabhi, declare that this thesis titled, 'Study of Efficient Mixed-Integer Planning for Autonomous Vehicles in Densely Cluttered Environments' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ____

Date: ____

"When Larry and Sergey founded Google Search, one of the things that struck me is that it was available for everyone to use. We deeply desire our services to work for everyone. And that inherently means we have to work with partners. That is the thesis underlying everything we do."

Sundar Pichai

SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY

Abstract

Bachelor of Technology
Electronics and Communication Engineering Department

Seminar report

by Mosam Sanjay Dabhi

In this work, we study the problem of planning safe and dynamically feasible trajectories through a densely cluttered environment. We devise an experimental approach to the design of smooth trajectories for quadrotor unmanned aerial vehicles (UAVs), which are free of collisions with obstacles along their entire length. To avoid the non-convex constraints normally required for obstacle-avoidance, we perform a mixed-integer optimization in which polynomial trajectories are assigned to convex regions which are known to be obstacle-free. Prior approaches have used the faces of the obstacles themselves to define these convex regions. The innovative approach brought about here is the use of Iterative Regional Inflation by Semidefinite Programming(IRIS) algorithm, a recently developed technique for greedy convex segmentation [1], to pre-compute convex regions of safe space. The aforementioned approach results in a substantially reduced number of integer variables, which improves the speed with which the optimization can be solved to its global optimum, even for tens or hundreds of obstacle faces. In addition, prior approaches have typically enforced obstacle avoidance at a finite set of sample or knot points. This study supports a technique based on sums-of-squares (SOS) programming that allows the user to ensure that the entire piecewise polynomial trajectory is free of collisions using convex constraints. To support this study, we demonstrate this technique in 2D and in 3D random environments and also in a cluttered indoor environment given by a dense pointcloud.

Acknowledgements

I acknowledge my department, 'Electronics and Communication Engineering Department' of S.V.N.I.T. for throughly supporting my work and I thank especially my seminar report advisor, 'Dr. J. N. Sarvaiya' for his valuable inputs and suggestions without which this report would not have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background motivation and need for this survey study

For autonomous vehicles traversing in real world environments, robust obstacle avoidance is a fundamental safety requirement. However, ensuring obstacle avoidance is complicated. We therefore wish to compute safe and optimal trajectories with respect to a cluttered environment. However, computing optimal trajectories through cluttered environments introduces additional computational complexity as the number of obstacles considered in the optimization increases. Advances in research have brought the quadrotor to a level of sophistication that is making it increasingly attractive for a variety of commercial applications like surveying and delivery. In order for these applications to become a reality, we now need algorithms that can deal reliably with environments that are substantially more cluttered than a laboratory setting. The most successful algorithms put forward to enable quadrotors to avoid obstacles often require an exponential increase in planning time with respect to the number of obstacles by, for example, introducing an integer variable for each face of obstacles in the environment. Most of these algorithms therefore start to perform poorly as the number of obstacles is increased beyond a modest handful of them. [2, 3].

For planning trajectories with obstacle avoidance in a known environment, generally we used to add non-convex constraints to an optimization for obstacle avoidance in order to ensure that our trajectory remains outside the set of ob- stacles. Such constraints prevent us from finding guarantees of completeness or global optimality in our program.

We can avoid some of the problems of non-convex constraints by adding a discrete component to optimization. So for that, if we model the non-convex set of obstacle-free states as union of finitely many convex regions, then we can perform a mixed-integer convex optimization in which the integer variables correspond to assignment of trajectory segments to convex regions. But, the approach mentioned is not without consequences, as even the addition of binary variables (con- strained to either take values 0 or 1) makes the problem from linear program into mixed-integer linear program, which is known to be NP-complete (as it is both NP and NP-hard). Earlier implementations typically used the faces of each obstacle to define the convex safe regions, which was used to encode obstacle avoidance for UAVs as a mixed integer linear program (MILP) by [3–6].

Recently, it has been suggested that the challenges of increasing obstacle density may be overcome by the novel Iterative Regional Inflation by semidifinite programming algorithm (IRIS) that segments space into obstacle-free regions. Planning paths through this segmentation can be accomplished using mixed-integer convex optimization, with the complexity growing with the much more manageable number of free-space segments instead of the number of obstacle faces. Hence, it can produce trajectories in environments containing many more obstacles than was previously demonstrated. Moreover the non-penetration constraint that the algorithm enforces along the entire path promise better performance with small obstacles than previous approaches that rely on sampling. We hypothesized that it would be possible to aggressively fly a small quadrotor through en- vironments containing greater number of obstacles than ever demonstrated before by leveraging IRIS, MISDPs, and model-based control approaches. This document presents experimental validation of this hypothesis and of the planning algorithm introduced in [7].

1.2 Related Work

A few planning algorithms have been successfully applied to planning for quadrotors in moderately crowded environments. The approach in [8] consisted of running RRT* in the entire space where the quadrotor might fly. The algorithm only expanded the randomly-exploring tree with straight paths in order to make its expansion more efficient. It there- fore resulted in a piece-wise linear collision-free trajectory. Finally, the algorithm computed a smooth trajectory using a quadratic program between each node along the path returned by RRT*. Although a very efficient approach, relying on sampling of space in order to check for collisions makes it potentially difficult to deal with very small obstacles like the strings used in our experiment. [2] demonstrated the use of mixed-integer quadratic programs in order to plan collision-free trajectories. In this particular work, the integer variable enforced non-penetration by making sure that the sampled location is on the collision-free side of at least one of the faces of each obstacle. The approach worked well in practice, but it suffered from having the number of integer variables grow rapidly with the number of obstacles. The technique once again cannot guarantee to generate collision- free trajectories between sample points, unlike our presented approach.

IRIS, the described choice of algorithm for convex segmentation of free-space, has previously been used in the context of collision-free footstep planning [9]. In this application, the algorithm generated obstacle-free regions in the robots configuration space. A mixed-integer program was then used to assign steps to each regions while simultaneously optimizing the pose of the robot. [7] then went on to demonstrate that we could formulate the problem of planning minimum-snap collision-free trajectories as a mixed-integer semidefinite program by using the obstacles-free regions returned by IRIS and by planning in differentially flat space. This document in part aims to provide experimental demonstrations for these results.

[10] demonstrated a feedback controller for aggressive flight inspired by the work of Hoffmann et al. [11]. In this approach, an off-board controller computed the position and velocity error of the quadrotor and an on-board attitude controller converted the associated desired center of mass acceleration to a desired attitude. Finally, [12] used time-varying LQR to demonstrate an aircraft with rotating wings executing a knife-edge maneuver in order to fly between two poles without colliding with either of them.

1.3 Modelling and System Identification

The proposed approach uses the differentially flat quadrotor model introduced in [13] and also used by others [8]. Generally, a system is said to be flat if there exists a set of output, in equal number to the number of inputs, such that all the states of the system

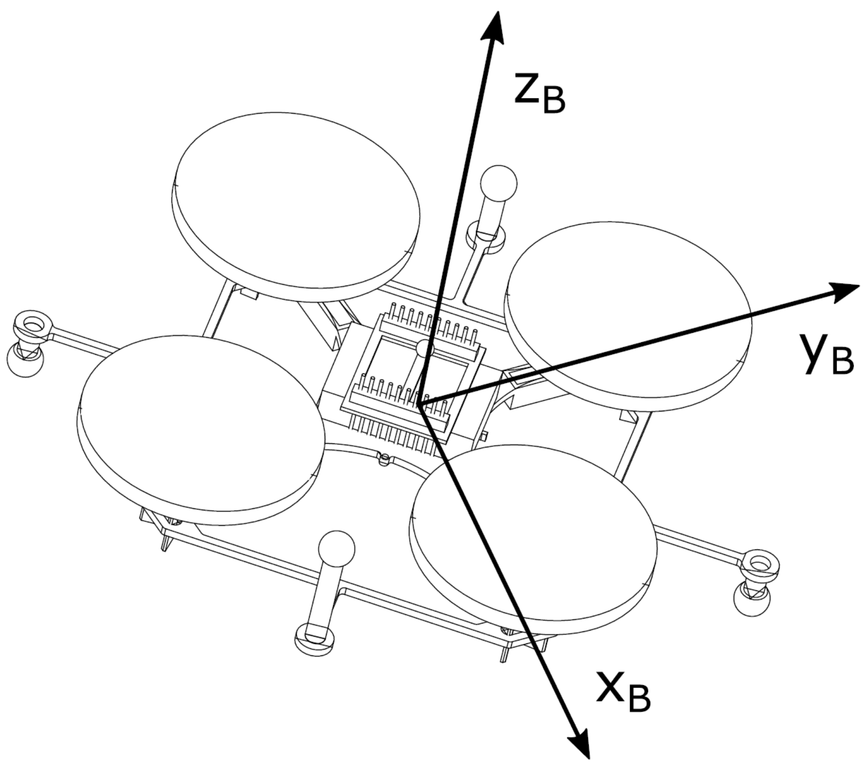


FIGURE 1.1: Model of the quadrotor used, the Crazyflie Nano Quadcopter, and the body frame used in the differentially flat model. $\boldsymbol{z_B}$ points in the same direction as propellers at all the time.

can be computed from these outputs (without integration). In our quadrotor model, the flat outputs are x, y, z, position and yaw.

The model used in the study consists of a single floating rigid body and four inputs. We define the inputs as the square of the angular velocities ω^2 of the motors on the quadrotor. A spinning propeller produces two forces on the quadrotor, namely lift and drag. Those forces are directly proportional to ω^2. Each input ω_i^2 can therefore be said to produce a certain linear force F_i on the center of mass as well as to create a moment M_i according to

$$F_i = k_f \omega_i^2, \tag{1.1}$$

$$M_i = k_m \omega_i^2. \tag{1.2}$$

We can then write the dynamics of the quadrotor:

$$m\ddot{r} = mg\boldsymbol{z_w} + (\sum_{i=1}^{4} F_i)\boldsymbol{z_b}, \tag{1.3}$$

$$\dot{\boldsymbol{\omega}} = I^{-1}(-\boldsymbol{\omega} \times I\boldsymbol{\omega} + W), \tag{1.4}$$

$$W = \begin{bmatrix} 0 & k_f L & 0 & -k_f L \\ -k_f L & 0 & k_f L & 0 \\ k_m & -k_m & k_m & -k_m \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}, \tag{1.5}$$

where m is the mass of the quadrotor, $\ddot{r}$ is the acceleration of its center of mass, $\boldsymbol{z_w}$ is a unit vector in the direction of gravity, $\boldsymbol{z_b}$ is a unit vector pointing in the same direction as the propellers (in the world frame), I is the inertia matrix of the quadrotor, L is the distance between each propeller and the center of mass of the quadrotor, and $\boldsymbol{\omega}$ is the angular velocity of the quadrotor in the body frame [8].

We solve the problem of identifying the parameters of this model with a two step approach. First, we directly or semi-directly measure each parameter of the model with a series of simple experiments. Second, we log flight data for a series of maneuvers using a motion capture system, and use an optimization-based algorithm to adjust those parameters. The k_f parameter was identified by measuring the thrust produced by the quadrotor placed upside-down on a scale and fitting the corresponding parameter. The k_m parameter can be measured by slowly increasing a pair of opposite motors (motors spinning in the same direction) and measuring the resulting angular velocities using the on-board gyroscope.

1.4 Avoiding Obstacles

The problem of obstacle avoidance is particularly chal- lenging because the set of points outside a closed, bounded obstacle is non-convex. As a result, we must generally add non-convex constraints to an optimization in order to ensure that our trajectory remains outside the set of obstacles. Such constraints generally prevent us from finding guarantees of completeness or global optimality in our program. [14]. We can avoid some of the problems of non-convex constraints by adding a discrete component to our optimization. If we model the non-convex set of obstacle-free states as the union of finitely many convex regions, then we can perform a mixed-integer convex optimization in which the integer variables correspond to the assignment of trajectory seg- ments to convex regions. This is not without consequences, since even the addition of binary variables (that is, integer variables constrained to take values of 0 or 1) turns linear programming into mixed-0, 1 linear programming, which is known to be NP-complete. [15]. However, excellent tools for solving a variety of mixed-integer convex problems have been developed in the past decade, and these tools can often find globally optimal solutions very efficiently for mixed- integer linear, quadratic, and conic problems [16–18]. These tools also offer some anytime capability, since they can be commanded to return a good feasible solution quickly or to spend more time searching for a provably global optimum.

Earlier implementations of mixed-integer obstacle avoid- ance have typically used the faces of the obstacles themselves to define the convex safe regions. The requirement that a point be outside all obstacles is converted to the requirement that, for each obstacle, the point must be on the outside of at least one face of that obstacle. For convex obstacles these conditions are equivalent [2]. This approach has been successfully used to encode obstacle avoidance for UAVs as a mixed integer linear program (MILP) by Schouwenaars [3], Richards [6], Culligan [4] and Hao [5]. Mellinger et al. add a quadratic cost function to turn the formulation into a mixed-integer quadratic program (MIQP) [2]. However, this approach requires separate integer variables for every face of every obstacle, which causes the mixed-integer formulation to become intractable with more than a handful of simple obstacles.

Instead, IRIS, a greedy tool for finding large convex regions of obstacle-free space is used here [1], to create a small number of convex regions to cover all or part of the free space. These regions can be seeded automatically based on heuristic methods or by an expert human operator. We then need only one integer variable for each region, rather than for each face of every obstacle. We have previously demonstrated this approach for mixed- integer footstep planning [19], and it is also evident from the proofs shown in the following chapters that it allows us to handle environments with tens or even hundreds of obstacle faces.

1.5 Safety of the Entire Trajectory

Through the proposed approach, we have the ability to ensure that the polynomial trajectory for the UAV is obstacle-free over its entire length, rather than at a series of sample points. Existing mixed-integer formulations have chosen only to enforce the

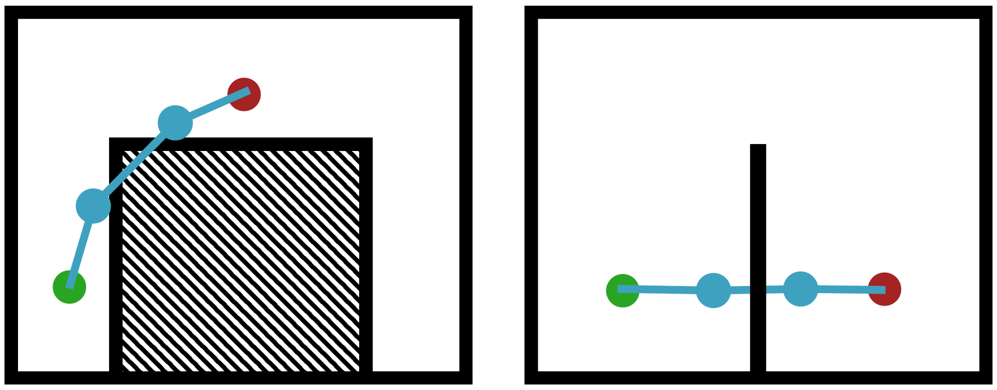


FIGURE 1.2: A piecewise linear trajectory between two points, with obstacle avoidance enforced only at 4 points along each trajectory. The continuous trajectory through those points may cut corners or pass through thin obstacles.

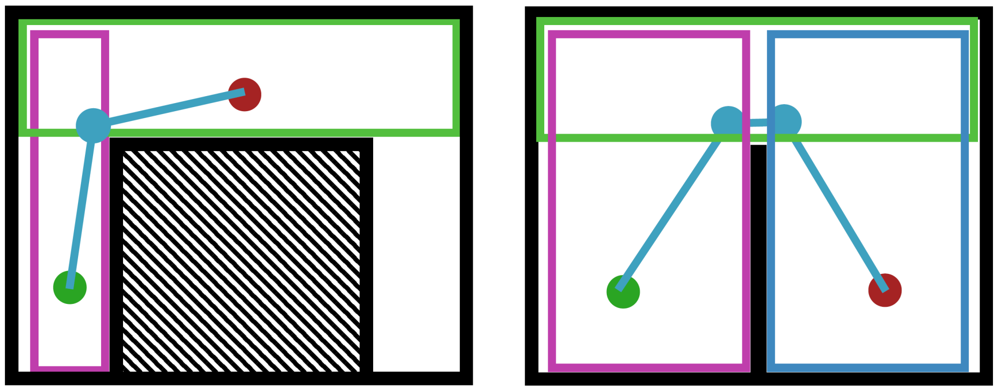


FIGURE 1.3: A trajectory in which each linear segment is required to remain en- tirely within one of the convex obstacle-free regions indicated by the colored boxes. This requirement ensures that the entire trajectory is collision-free.

obstacle-avoidance constraint at a finite set of points [2, 3, 20]. This can result in the path between those sample points cutting the corners of obstacles, or, more dangerously, passing through very thin obstacles, as shown in Fig. 1.2. As noted by [21], the severity of the corner cuts can be reduced by increasing the number of sample points and limiting the total distance between adjacent samples, but this also increases the complexity of the optimization problem. Mellinger approaches this problem by requiring that the bounding boxes of the UAV at adjacent sample points must overlap [2]. This is sufficient to ensure that the UAV never passes entirely through an obstacle, but it does not necessarily prevent corner cutting.

Representing the environment with convex safe regions allows us to completely eliminate the cutting of corners. If we treat the problem as one of assigning entire segments of trajectories, rather than points, to the safe regions, then we can create a fully collision-free trajectory. For piecewise linear trajectories, this is simply a matter of ensuring that, for each linear segment, both endpoints must be contained within the same convex safe region, as shown in Fig. 1.3. This decision weakens our claim of optimality, since it requires the breakpoints between trajectory segments to occur in the inter- section of two convex regions, but it results in a formulation that can be solved exactly with mixed-integer programming. We enforce the constraint that each polynomial lie within a convex region using a sums-of-squares (SOS) approach. In this way, collision-free trajectories can be generated using piecewise polynomials of arbitrary degree. Here we show that, for trajectory segments defined by polynomials, we can exactly enforce the constraint that each segment lies inside a convex region by solving a small semi-definite program(SDP).

These polytopes, generated from IRIS must be given as an ordered union of only pairwise adjacent sets, but the trajectories are guaranteed to be contained within the polytopes by construction. Since the polytopes must be pairwise adjacent, they must be laid out along a single path from start to goal by some other planning procedure, and the resulting trajectories may not leave this path. On the other hand, by performing our mixed-integer optimization, we are able to explore arbitrarily connected polytopes which may admit many different possible paths through them.

Chapter 2

Technical Approach

The proposed trajectory planning problem has three components:

1. Generating convex safe regions
2. Assigning trajectory segments to those convex regions and
3. Optimizing the trajectories while ensuring that they remain fully within the safe regions.

Step (1) as a pre- processing stage, then (2) and (3) are performed simultaneously in a single mixed-integer convex optimization, which can be solved to global optimality.

2.1 Generating Convex Regions of Safe Space

The user's ability to efficiently segment the space into convex regions relies on our recent development of IRIS (Iterative Regional Inflation by Semidefinite programming) [1]. IRIS alternates between two convex optimizations that (A) find a set of hyperplanes which separate some ellipsoid from the obstacles and (B) find the largest-volume ellipsoid within those hyperplanes. Given an initial seed point in space, around which the first ellipsoid is constructed, IRIS grows the ellipsoid greedily at every iteration until it reaches a local fixed point. The final set of separating hyperplanes forms a (convex) polytope, which we take as our convex region of obstacle-free space. Additional runs of IRIS with different seed points produce additional obstacle-free regions.

Our initial applications of IRIS to the footstep planning problem for walking robots relied on a human user to choose the seed points for IRIS [19]. Human input was valuable for that problem, since the choice of seed locations allowed an expert operator to provide high-level input such as which surfaces should be used for stepping. Manually seeded regions are, of course, also possible in the case of an aerial vehicle, and we expect that having an expert operator choose the location of the seeds might be beneficial when the environment is largely static and known beforehand. However, this requirement is overly restrictive in the general case.

2.2 Searching over Assignments of Polynomials to Regions

We encode the assignment of each polynomial piece of the trajectory to a safe region using a matrix of binary integer variables $H_{r,j} \in \left\{0, 1\right\}^{R \times N}$, where R is the number of regions and N is the number of polynomial trajectory pieces. The polynomial trajectory pieces are labeled as $P_j(t)$ and the convex regions as $\mathcal{G}_r$. Thus, we have:

$$H_{r,j} \Rightarrow p_j(t) \in \mathcal{G}_r \qquad \forall t \in [0, T_j] \tag{2.1}$$

The range of $[0, 1]$ for time is chosen arbitrarily for simplicity in this discussion, but any desired time span can be chosen when constructing the problem. The actual time spent executing each trajectory segment can also be adjusted as a post- processing step by appropriately scaling the coefficients.

Ensuring that polynomial j is collision-free is expressed with a linear constraint on H:

$$\sum_{r=1}^{R} H_{r,j} = 1 \tag{2.2}$$

Note that the regions do overlap, so it is possible for a polynomial to simultaneously exist within multiple regions $\mathcal{G}_r$. Such a case is allowed by the proposed formulation, since the implication in 2.1 is one-directional (so polynomial $P_j(t)$ being contained in $\mathcal{G}_r$ does not necessarily require that $H_{r,j} = 1$). We show in next section that the constraint $P_j(t) \in \mathcal{G}_r \forall t \in [0, 1]$ is convex, and we can use a standard big-M formulation [22] to convert the implication in 2.1 to a linear form.

2.3 Restricting a Polynomial to a Polytope

The trajectories are represented in n dimensions as piecewise polynomials of degree d in a single variable, t. Each segment j of the trajectory is parameterized by $d+1$ vectors of coefficients $C_{j,k} \in \mathbb{R}^n$ of a set of polynomial basis functions, $\Phi_1(t), ..., \Phi_{d+1}(t)$. For each segment j, the trajectory can be evaluated as

$$P_j(t) = \sum_{k=1}^{d+1} C_{j,k}\Phi_k(t) \qquad t \in [0,1] \tag{2.3}$$

R convex regions of safe space are restricted to be polytopes, so for each region $r \in 1, ..., R$, some $A_r \in \mathbb{R}^{m\times n}$ and $b_r \in \mathbb{R}^m$ and the constraint that

$$A_r P_j(t) \leq b_r \tag{2.4}$$

if $H_{r,j}$ is set to 1. To ensure that the trajectory remains entirely within the safe region, 2.4 must hold $\forall t \in [0,1]$.

$$A_r \sum_{k=1}^{d+1} C_{j,k}\Phi_k(t) \leq b_r \qquad \forall t \in [0,1] \tag{2.5}$$

Eq. 2.5 consists of m constraints of the form

$$a_{r,l}^T \sum_{k=1}^{d+1} C_{j,k}\Phi_k(t) \leq b_{r,l} \qquad \forall t \in [0,1] \tag{2.6}$$

where

$$A_r = \begin{bmatrix} a_{r,1}^T \\ a_{r,2}^T \\ . \\ . \\ . \\ a_{r,m}^T \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} b_{r,1} \\ b_{r,2} \\ . \\ . \\ . \\ b_{r,m} \end{bmatrix} \tag{2.7}$$

Redistributing the terms in 2.6 to get

$$\sum_{k=1}^{d+1} (a_{r,l}^T C_{j,k})\Phi_k(t) \leq b_{r,l} \qquad \forall t \in [0,1] \tag{2.8}$$

and thus

$$q(t) := b_{r,l} - \sum_{k=1}^{d+1} (a_{r,l}^T C_{j,k}) \Phi_k(t) \geq 0 \qquad \forall t \in [0,1] \tag{2.9}$$

The condition that $q(t) \geq 0 \quad \forall t \in [0,1]$ holds if and only if $q(t)$ can be written as

$$\begin{aligned} q(t) &= t\sigma_1(t) + (1-t)\sigma_2(t) \qquad &\text{if } d \text{ is odd} \\ &= \sigma_1(t) + t(1-t)\sigma_2(t) \qquad &\text{if } d \text{ is even} \end{aligned} \tag{2.10}$$

where the most important condition for viability of 2.10 is: $\sigma_1(t), \sigma_2(t)$ are sums of squares, where $\sigma_1(t)$ and $\sigma_2(t)$ are polynomials of degree $d-1$ if d is odd and of degree d and $d-2$ if d is even [23]. The condition that $\sigma_1(t), \sigma_2(t)$ are sums of squares requires that each can be decomposed into a sum of squared terms, which is a necessary and sufficient condition for non-negativity for polynomials of a single variable [23]. The coefficients of the polynomials σ_1 and σ_2 are additional decision variables in our optimization, subject to linear constraints to enforce 2.10. The sum-of-squares constraints can be rep- resented in general with a semidefinite program [24]. The problem of assigning the trajectories to safe regions is thus a mixed-integer semidefinite program (MISDP). This class of problems can be solved to global optimality using, for example, the Yalmip branch-and-bound solver or with a semidefinite programming solver like Mosek [17]. Successful validation for the above formulation is applied to polynomials of degree 1, 3 and 5. For polynomials of degree 7 and higher, due to numerical difficulties which often prevented Mosek from solving the semidefinite program, more numerically stable exact reductions for lower degree polynomials is proposed.

For polynomials of degree 1, σ_1 and σ_2 are constants, and the condition of being the sums of squares reduces to linear constraints

$$\sigma_1 \geq 0, \sigma_2 \geq 0, \tag{2.11}$$

which reduces the problem to a mixed-integer quadratic program (MIQP), given our quadratic objective function. If the polynomials are of degree 3, then $\sigma_i(t)$ is a quadratic polynomial:

$$\sigma_i(t) = \beta_1 + \beta_2(t) + \beta_3(t)^2. \tag{2.12}$$

Using the standard sum-of-squares approach, we rewrite $\sigma_i(t)$ as

$$\sigma_i(t) = \begin{bmatrix} 1 & t \end{bmatrix} \begin{bmatrix} \beta_1 & \beta_2/2 \\ \beta_2/2 & \beta_3 \end{bmatrix} \begin{bmatrix} 1 \\ t \end{bmatrix} \tag{2.13}$$

The condition that $\sigma(t)$ is SOS is equivalent to the matrix of coefficients in 2.13 being positive semi-definite:

$$\begin{bmatrix} \beta_1 & \beta_2/2 \\ \beta_2/2 & \beta_3 \end{bmatrix} \geq 0, \tag{2.14}$$

which is in turn equivalent to the following rotated second- order cone constraint:

$$\beta_2^2 - 4\beta_1\beta_3 \leq 0 \tag{2.15}$$

$$\beta_1, \beta_3 \geq 0 \tag{2.16}$$

Thus the problem of assigning degree-3 poly- nomials to convex regions as a mixed-integer second-order cone problem (MISOCP) can be solved effectively with Mosek [17].

2.4 Choosing Objective Function

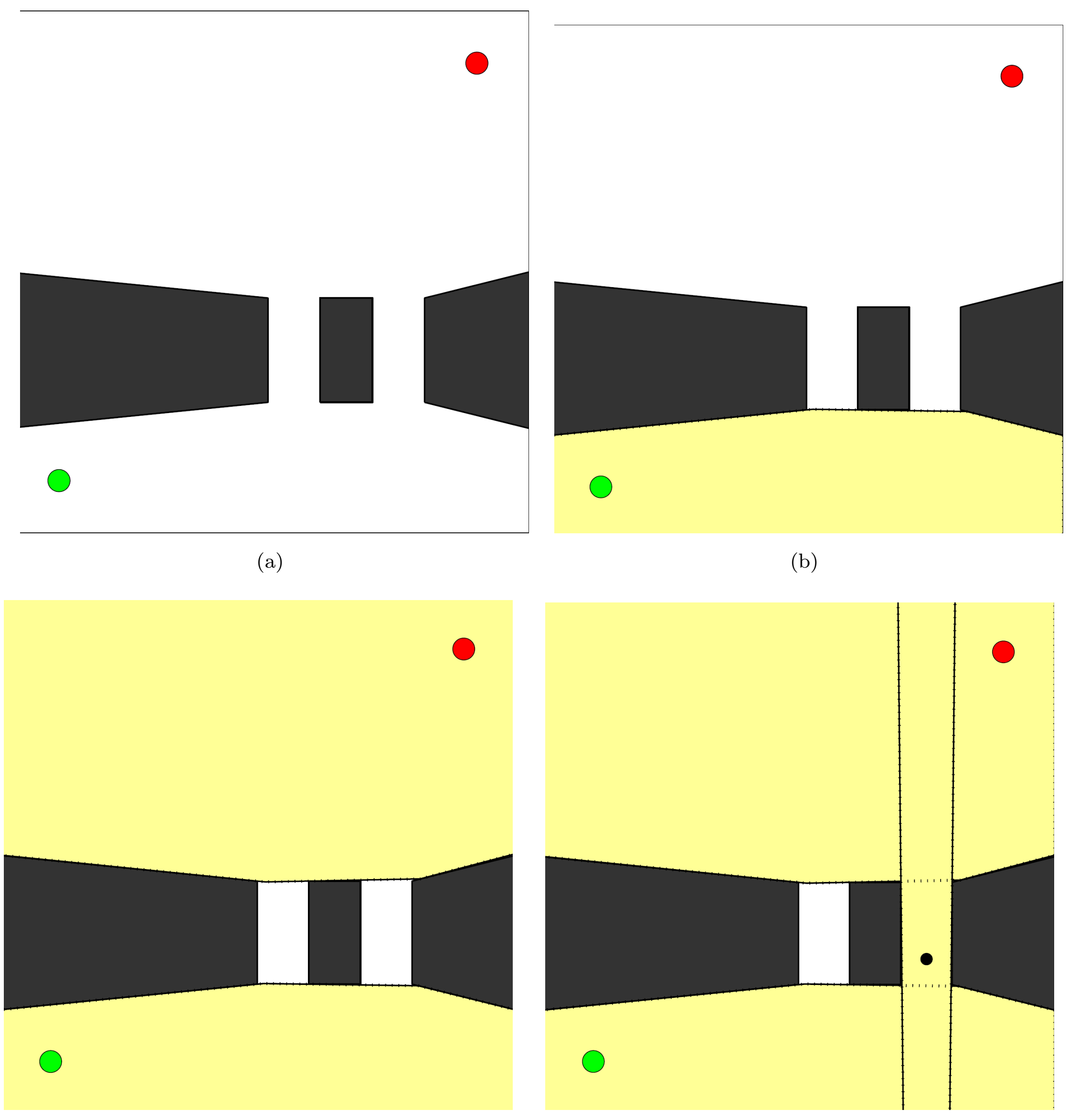
If $P_j(t)$ is of degree $d \geq 4$, then according to Mellinger et al. who relate the snap (that is, the fourth derivative of position) to the control inputs of a quadrotor, and thus an objective of the form:

$$\text{minimize} \quad \sum_{j=1}^{N} \int_0^1 \left\| \tfrac{\mathrm{d}^4}{\mathrm{d}t^4} P_j(t) \right\| \mathrm{d}t \tag{2.17}$$

If $P_j(t)$ is of degree $d \geq 4$ then we may do likewise, resulting in a convex quadratic objective on the coefficients of the P_j. Demonstrating this objective function in operation with 5th degree polynomials in Fig. 2.1 and 2.2.

However, to reduce our problem to a MISOCP and improve the numerical stability of the solver, is is found that it is beneficial to restrict ourselves to 3rd-degree polynomials and thus piecewise constant jerk. Our objective is:

$$\text{minimize} \quad \sum_{j=1}^{N} \int_0^1 \left\| \tfrac{\mathrm{d}^3}{\mathrm{d}t^3} P_j(t) \right\| \mathrm{d}t \tag{2.18}$$



(a) (b) (c) (d)

FIGURE 2.1: Solving a simple environment with IRIS regions. A random environment is constructed with obstacles and a start and goal pose (a), then generate IRIS regions around the start (b) and goal (c). Next, a point is identified far from the existing set of obstacles and IRIS regions and a new region is generated from the seed point there (d)

which is likewise convex and quadratic in the coefficients of the P_j .Linear constraints are added on the position, velocity, and acceleration of each trajectory piece to ensure that they are continuous from one polynomial piece to the next. Additional linear equality constraints require that the position, velocity, and acceleration of the beginning of the first trajectory piece and the end of the last piece match our desired initial and final states, thereby giving a perfect smooth trajectory.

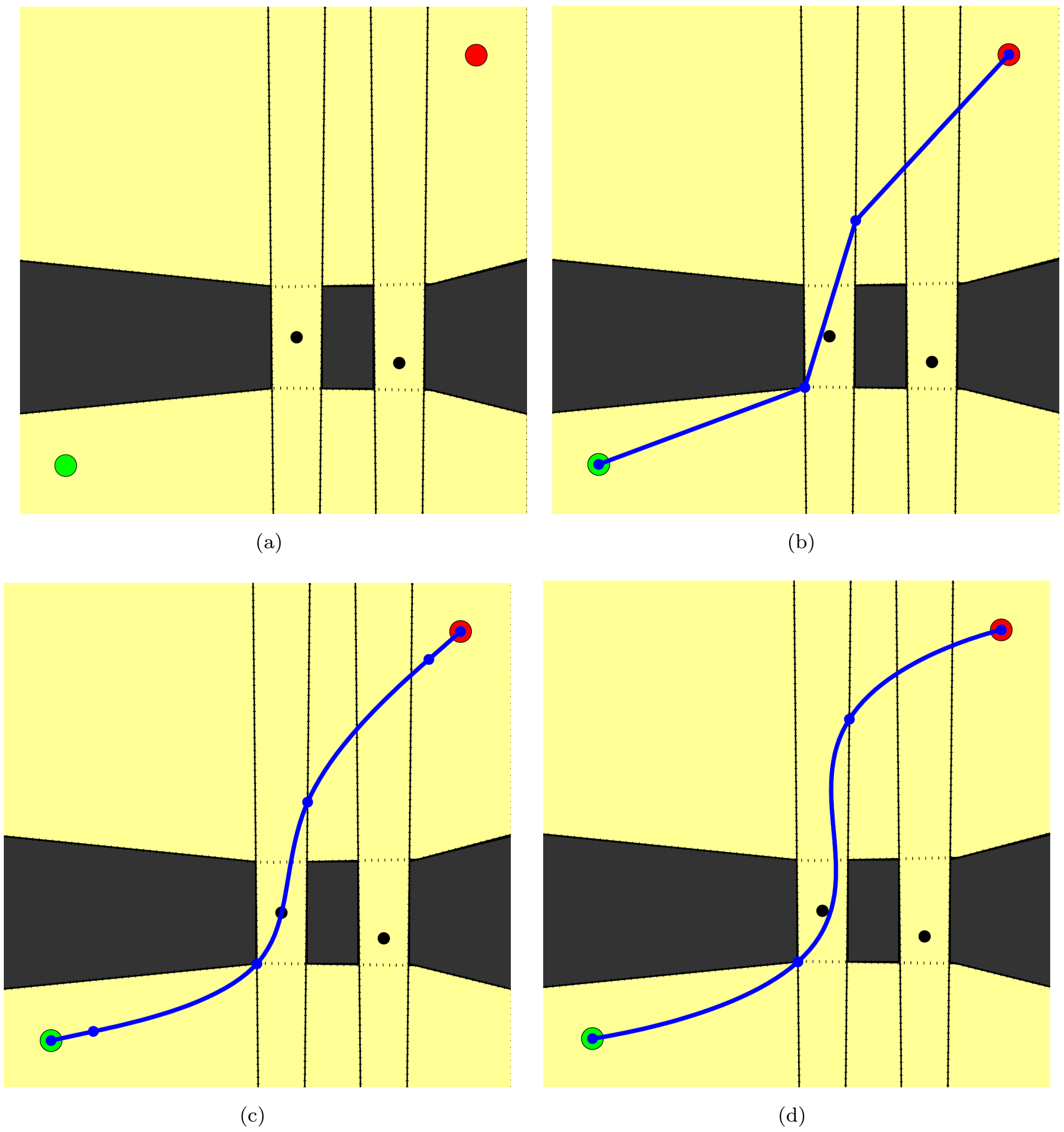


FIGURE 2.2: The above process through which we obtained (d) is repeated until we have 4 regions (e). Finally, we solve for trajectories of 1^{st}-degree polynomials minimizing squared velocity in 0.1s (f), 3^{rd}-degree polynomials minimizing squared jerk in 0.3s (g), and 5^{th}-degree polynomials minimizing squared snap in 1.0s (h). All trajectories lie entirely within the convex regions shown.

2.5 Handling Lower-Degree Trajectories

Even if the mixed-integer optimization is done over the numerically easier degree 3 polynomials, we can post-process the resulting trajectories in order to successfully use the differential flatness of the system to derive the full state and input. A piecewise degree-3 trajectory has a piecewise constant 3^{rd} derivative. It thus has delta functions for its 4^{th} derivative, which Mellinger relates directly to the rotor thrusts of the UAV [13]. Since this is clearly undesirable, according the proposed approach we proceeded as follows: First, we ran the MISOCP to optimize our degree-3 polynomials and assign them to convex safe regions. Next, we fixed the resulting assignment of trajectories to safe regions and then re-ran the optimization for polynomials of degree 5 or higher while minimizing the squared norm of the snap. Since all of the integer variables are fixed, we no longer have a mixed-integer problem but instead a single semidefinite program, which can be solved very efficiently.

2.6 Complete Formulation

Our optimization problem can be written as follows for a trajectory of N piecewise 3^{rd}-degree polynomials:

$$\min_{P,H,\sigma} \quad \sum_{j=1}^{N} \left\| \tfrac{\mathrm{d}^3}{\mathrm{d}t^3} P_j(t) \right\| \tag{2.19}$$

subject to:

$$\begin{aligned}
&P_1(0) = x_0, \quad P_N(1) = x_f, \quad P_j(1) = P_{j+1}(0)\\
&\dot{P}_1(0) = \dot{x_0}, \quad \dot{P}_N(1) = c\dot{x_f}, \quad \dot{P}_j(1) = \dot{P}_{j+1}(0)\\
&\ddot{P}_1(0) = \ddot{x_0}, \quad \ddot{P}_N(1) = \ddot{x_f}, \quad \ddot{P}_j(1) = \ddot{P}_{j+1}(0)
\end{aligned}$$

$$\forall j \in \Big\{1, \quad . \quad . \quad . \quad , N\Big\} \tag{2.20}$$

where $\sigma_{l,j,1}(t), \sigma_{l,j,2}(t)$ are sums of squares

$$\begin{aligned}
&H_{r,j} \implies\\
&\quad b_{r,m} - A_{r,m}^T p_j(t) = t\sigma_{j,m,1}(t) + (T_j - t)\sigma_{j,m,2}(t)
\end{aligned}$$

$$\textstyle\sum_{r=1}^{R} H_{r,j} = 1 \quad \forall j \in 1, ..., N$$

$$H_{r,j} \in 0, 1 \tag{2.21}$$

where x_0 , $\dot{x_0}$, $\ddot{x_0}$ are the initial position, velocity, and acceleration of the vehicle and x_f , $\dot{x_f}$, $\ddot{x_f}$ are the final values. All of the above conditions are linear constraints on the coefficients C and β and the matrix H, except the condition that σ_1 and σ_2 are sums of squares, which is a rotated second-order cone constraint.

Chapter 3

Conclusion

We studied a new method for optimal trajectory planning around obstacles which ensures that the entire path is collision-free, rather than enforcing obstacle avoidance only at a set of sample points. This method is formulated as a mixed-integer convex program and can be directly used with the mixed-integer obstacle avoidance approach which is already common in the field. Performance of our approach can be significantly improved by pre-computing convex regions of safe space with IRIS, a tool for greedy convex segmentation, which can allow us to solve for trajectories even in very cluttered environments.

3.1 Limitations

By requiring that each polynomial trajectory piece lie entirely within one convex safe region, we disallow trajectories which may not intersect the obstacles but which pass through several safe regions. Our claims of global optimality are also limited to trajectories which obey this restriction. This problem can be alleviated by increasing the number of trajectory segments so that each segment can be assigned to a single safe region, but doing so increases the complexity of the mixed-integer program. Successful trajectory generation is also dependent on the particular set of convex regions which are generated. In some environments, automatically finding regions at points far from the obstacles was sufficient, but as the environment becomes more complex, we may require a more intelligent method of selecting the seed points at which the IRIS algorithm begins. Input from a human operator can be extremely helpful, a human operator indicated the interiors of the window and doorway as salient points at which to generate convex regions, which allowed a feasible trajectory to be found with less time spent blindly searching for good region locations.

Finally, in order to ensure a smooth control input, we may wish to constrain the derivatives of the snap of the trajectory, which will require polynomials of degree 5 or higher. This will require a more careful approach to ensure the numerical stability of the mixed-integer semidefinite program. We are not yet able to reliably solve these high-order problems using Mosek without encountering numerical difficulties. The choice of basis functions Φ in Eq. 2.3 is likely to be a significant factor in the numerical stability of the solver [2].

3.2 Future Work

In the future, intention to explore additional constraints and objectives, such as way-points in space which must be visited en route from the start to the goal. Plans to study investigate more effective heuristics for choosing the seed points for the convex safe regions, including seeding regions based on the results of a sample-based motion planner like RRT. Finally, we plan to study these trajectories into the real world with their stabilization and execution on hardware.

Appendix A

An Appendix

Add appendices here

Bibliography

[1] Robin Deits and Russ Tedrake. Computing large convex regions of obstacle-free space through semidefinite programming. In *Algorithmic Foundations of Robotics XI*, pages 109–124. Springer, 2015.

[2] Daniel Mellinger, Alex Kushleyev, and Vijay Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 477–483. IEEE, 2012.

[3] Tom Schouwenaars, Bart De Moor, Eric Feron, and Jonathan How. Mixed integer programming for multi-vehicle path planning. In *Control Conference (ECC), 2001 European*, pages 2603–2608. IEEE, 2001.

[4] Kieran Forbes Culligan. *Online trajectory planning for uavs using mixed integer linear programming.* PhD thesis, Massachusetts Institute of Technology, 2006.

[5] Yongxing Hao, Asad Davari, and Ali Manesh. Differential flatness-based trajectory planning for multiple unmanned aerial vehicles using mixed-integer linear programming. 2005.

[6] Arthur Richards, John Bellingham, Michael Tillerson, and Jonathan How. Coordination and control of multiple uavs.

[7] Robin Deits and Russ Tedrake. Efficient mixed-integer planning for uavs in cluttered environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–49. IEEE, 2015.

[8] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for quadrotor flight.

[9] Robin LH Deits. *Convex segmentation and mixed-integer footstep planning for a walking robot.* PhD thesis, Massachusetts Institute of Technology, 2014.

[10] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, page 0278364911434236, 2012.

[11] Gabriel M Hoffmann, Steven L Waslander, and Claire J Tomlin. Quadrotor helicopter trajectory tracking control. In *AIAA guidance, navigation and control conference and exhibit*, pages 1–14, 2008.

[12] Andrew J Barry. *Flying between obstacles with an autonomous knife-edge maneuver.* PhD thesis, Citeseer, 2012.

[13] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525. IEEE, 2011.

[14] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[15] Richard M Karp. Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008*, pages 219–241. Springer, 2010.

[16] Gurobi Optimization. Gurobi optimizer reference manual version 5.6, 2014.

[17] Mosek ApS. The mosek optimization software. *Online at http://www. mosek. com*, 2014.

[18] IBM ILOG. Cplex optimizer. *En ligne]. Available: http://www-01. ibm. com/software/commerce/optimization/cplex-optimizer*, 2010.

[19] Robin Deits and Russ Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 279–286. IEEE, 2014.

[20] Arthur Richards and Jonathan P How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, volume 3, pages 1936–1941. IEEE, 2002.

[21] John Saunders Bellingham. *Coordination and control of uav fleets using mixed-integer linear programming*. PhD thesis, Citeseer, 2002.

[22] Johan Lofberg. Yalmip: A toolbox for modeling and optimization in matlab. In *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, pages 284–289. IEEE, 2004.

[23] PA Parrilo. sums of squares and semidefinite programming, 2006.

[24] Victoria Powers and Thorsten Wörmann. An algorithm for sums of squares of real polynomials. *Journal of pure and applied algebra*, 127(1):99–104, 1998.