2015 IEEE International Conference on Robotics and Automation (ICRA)
Washington State Convention Center
Seattle, Washington, May 26-30, 2015

Efficient Mixed-Integer Planning for UAVs in Cluttered Environments

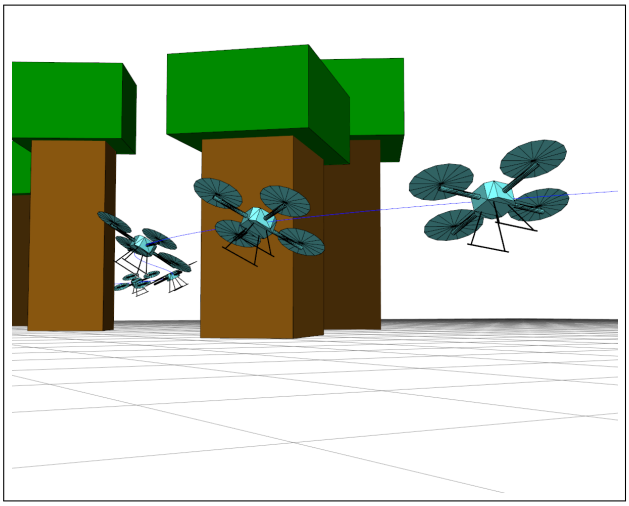
Robin Deits[1] and Russ Tedrake[2]

***Abstract*— We present a new 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. We instead use IRIS, a recently developed technique for greedy convex segmentation [1], to pre-compute convex regions of safe space. This 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. We introduce a technique based on sums-of-squares (SOS) programming that allows us to ensure that the entire piecewise polynomial trajectory is free of collisions using convex constraints. We demonstrate this technique in 2D and in 3D using a dynamical model in the Drake toolbox for MATLAB [2].**

I. INTRODUCTION

We consider the problem of planning a feasible trajectory for a quadrotor UAV from an initial state to a goal state while avoiding obstacles. Rather than explicitly considering the full state of the quadrotor, including its pose, velocity, and rotor thrusts, we rely on the recent results from Mellinger and Kumar, who demonstrated that the quadrotor system is *differentially flat* with respect to the 3D position and yaw of the vehicle's center of mass [3]. That is, the entire state of the system can be expressed as a function of the instantaneous value of the x, y, z positions of the CoM and the yaw of the vehicle, along with their derivatives. As a result, any smooth trajectory with sufficiently bounded derivatives can be executed by the quadrotor. This means that we are free to design smooth trajectories of the position and yaw of the vehicle's center of mass without explicitly considering the dynamics.

We define a trajectory as a piecewise polynomial function in time with vector-valued coefficients, mapping time to position in 2D or 3D space. We choose the degree of the polynomials and the number of pieces offline. Our optimization problem is a matter of choosing the coefficients of each polynomial in order to ensure that the trajectory reaches a desired goal state, avoids collisions, and satisfies an objective function of our choosing.



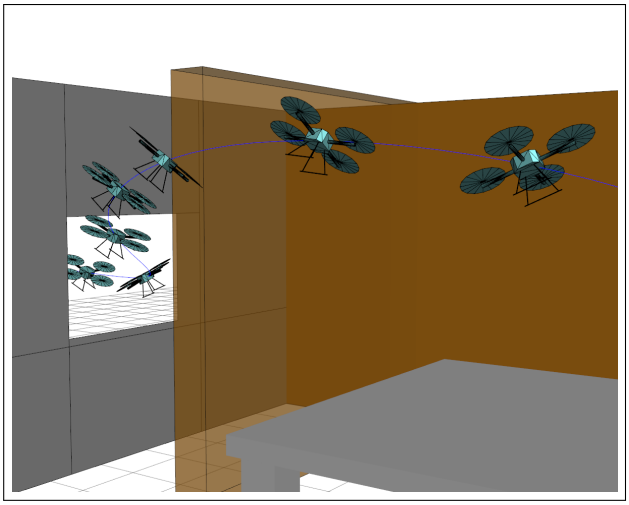


Fig. 1. Fully collision-free trajectories for a quadrotor UAV in a simulated forest and office environment, showing 1.6s of planned trajectory execution.

A. Avoiding Obstacles

The problem of obstacle avoidance is particularly challenging 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 [4]. We can avoid some of the problems of non-convex constraints by adding a discrete component to our optimization. If

This work was supported by the Fannie and John Hertz Foundation, the MIT Energy Initiative, and the MIT Computer Science and Artificial Intelligence Lab.

[1]Robin Deits is with the Computer Science and Artificial Intelligence Lab, MIT, Cambridge, MA 02139, USA rdeits@csail.mit.edu

[2]Russ Tedrake is with the Faculty of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139, USA russt@csail.mit.edu

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 segments 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 [5]. 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 [6], [7], [8]. 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 avoidance 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 [9]. This approach has been successfully used to encode obstacle avoidance for UAVs as a mixed integer linear program (MILP) by Schouwenaars [10], Richards [11], Culligan [12], and Hao [13]. Mellinger et al. add a quadratic cost function to turn the formulation into a mixed-integer quadratic program (MIQP) [9]. 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, we use our recent work developing IRIS, a greedy tool for finding large convex regions of obstacle-free space [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 [14], and we show in Sect. III that it allows us to handle environments with tens or even hundreds of obstacle faces.

B. Safety of the Entire Trajectory

Our second contribution is 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 obstacle-avoidance constraint at a finite set of points [9], [10], [15]. 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. 2. As noted by Bellingham [16], 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

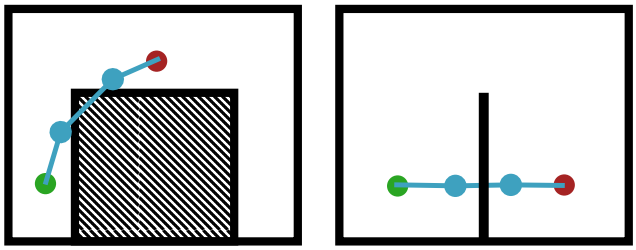


Fig. 2. A piecewise linear trajectory between two points, with obstacle avoidance enforced only at 4 points along each trajectory. The continuous trajectory through those points may cut corners or pass through thin obstacles.

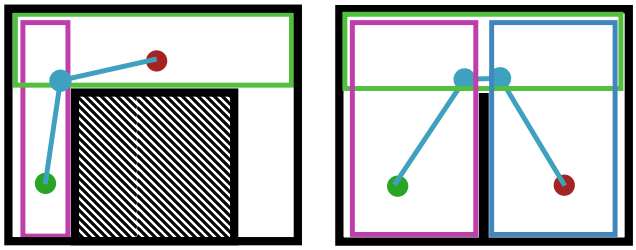


Fig. 3. A trajectory in which each linear segment is required to remain entirely within one of the convex obstacle-free regions indicated by the colored boxes. This requirement ensures that the entire trajectory is collision-free.

problem. Mellinger approaches this problem by requiring that the bounding boxes of the UAV at adjacent sample points must overlap [9]. 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. 3. This decision weakens our claim of optimality, since it requires the breakpoints between trajectory segments to occur in the intersection 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).

This is somewhat similar to work by Flores, who uses non-uniform rational B-splines to generate trajectories which are completely contained within convex polytopes [17]. These polytopes 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.

II. TECHNICAL APPROACH

The trajectory planning problem we propose 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. We perform step (1) as a pre-processing stage, then solve (2) and (3) *simultaneously* in a single mixed-integer convex optimization, which can be solved to global optimality.

A. Generating Convex Regions of Safe Space

Our 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 [14]. 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.

For that reason, we have developed a simple heuristic for automatically seeding IRIS regions with no human input. First, the space is discretized into a coarse grid. We then choose the grid cell which maximizes the distance to the nearest obstacle and seed one IRIS region at that point. We then repeat, choosing a new seed which maximizes the distance to the nearest obstacle *or* existing IRIS region. Currently, we terminate the algorithm when a pre-defined number of regions have been generated, but we intend to explore other stopping criteria, such as a threshold on the volume of the space which is currently unoccupied by obstacles or regions. Automatic seeding of IRIS regions is shown in Sect. III.

B. 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 $H \in \{0,1\}^{R \times N}$, 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 G_r. Thus, we have:

$$H_{r,j} \implies P_j(t) \in G_r \quad \forall t \in [0,1] \tag{1}$$

We arbitrarily choose the range of $[0,1]$ 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:

$$\sum_{r=1}^{R} H_{r,j} = 1 \tag{2}$$

Note that we allow the regions to overlap, so it is possible for a polynomial to simultaneously exist within multiple regions G_r. Such a case is allowed by our formulation, since the implication in (1) is one-directional (so polynomial $P_j(t)$ being contained in G_r does not necessarily require that $H_{r,j} = 1$).

We show in Sect. II-C that the constraint $P_j(t) \in G_r \quad \forall t \in [0,1]$ is convex, and we can use a standard big-M formulation [18] to convert the implication in (1) to a linear form.

C. Restricting a Polynomial to a Polytope

We represent our trajectories 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, $\Phi_1(t), \ldots, \Phi_{d+1}(t)$. For each segment j, the trajectory can be evaluated as

$$P_j(t) = \sum_{k=1}^{d+1} C_{j,k} \Phi_k(t) \quad t \in [0,1] \tag{3}$$

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

$$A_r \, P_j(t) \leq b_r \tag{4}$$

if $H_{r,j}$ is set to 1. To ensure that the trajectory remains entirely within the safe region, we require that (4) hold for all $t \in [0,1]$

$$A_r \sum_{k=1}^{d+1} C_{j,k} \Phi_k(t) \leq b_r \quad \forall t \in [0,1]. \tag{5}$$

Eq. 5 consists of m constraints of the form

$$a_{r,\ell}^{\top} \sum_{k=1}^{d+1} C_{j,k} \Phi_k(t) \leq b_{r,\ell} \quad \forall t \in [0,1] \tag{6}$$

where

$$A_r = \begin{bmatrix} a_{r,1}^\top \\ a_{r,2}^\top \\ \vdots \\ a_{r,m}^\top \end{bmatrix} \text{ and } b = \begin{bmatrix} b_{r,1} \\ b_{r,2} \\ \vdots \\ b_{r,m} \end{bmatrix}. \tag{7}$$

We can redistribute the terms in (6) to get

$$\sum_{k=1}^{d+1} (a_{r,\ell}^\top C_{j,k}) \Phi_k(t) \leq b_{r,\ell} \quad \forall t \in [0,1] \tag{8}$$

and thus

$$q(t) := b_{r,\ell} - \sum_{k=1}^{d+1} (a_{r,\ell}^\top C_{j,k}) \Phi_k(t) \geq 0 \quad \forall t \in [0,1]. \tag{9}$$

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

$$q(t) = \begin{cases} t\sigma_1(t) + (1-t)\sigma_2(t) & \text{if } d \text{ is odd} \\ \sigma_1(t) + t(1-t)\sigma_2(t) & \text{if } d \text{ is even} \end{cases} \tag{10}$$

$$\sigma_1(t),\ \sigma_2(t) \text{ are sums of squares} \tag{11}$$

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 [19]. 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 nonnegativity for polynomials of a single variable [19]. The coefficients of the polynomials σ_1 and σ_2 are additional decision variables in our optimization, subject to linear constraints to enforce (10). The sum-of-squares constraints in (11) can be represented in general with a semidefinite program [20]. 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 [21] combined with a semidefinite programming solver like Mosek [7] or using the dedicated SDP package developed by Mars and Schewe [22]. We have successfully applied this formulation to polynomials of degree 1, 3, 5, and 7. For polynomials of degree 7 and higher, we experienced numerical difficulties which often prevented Mosek from solving the semidefinite program. As a result, we have developed more numerically stable exact reductions for lower degree polynomials.

For polynomials of degree 1, σ_1 and σ_2 are constants, and the condition in (11) reduces to linear constraints

$$\sigma_1 \geq 0,\ \sigma_2 \geq 0, \tag{12}$$

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:

$$\sigma_i(t) = \beta_1 + \beta_2 t + \beta_3 t^2. \tag{13}$$

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

$$\sigma_i(t) = \begin{bmatrix} 1 & t \end{bmatrix} \begin{bmatrix} \beta_1 & \frac{\beta_2}{2} \\ \frac{\beta_2}{2} & \beta_3 \end{bmatrix} \begin{bmatrix} 1 \\ t \end{bmatrix}. \tag{14}$$

The condition that $\sigma(t)$ is SOS is equivalent to the matrix of coefficients in (14) being positive semi-definite:

$$\begin{bmatrix} \beta_1 & \frac{\beta_2}{2} \\ \frac{\beta_2}{2} & \beta_3 \end{bmatrix} \succeq 0, \tag{15}$$

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

$$\beta_2^2 - 4\beta_1\beta_3 \leq 0 \tag{16}$$

$$\beta_1, \beta_3 \geq 0 \tag{17}$$

We can thus write the problem of assigning degree-3 polynomials to convex regions as a mixed-integer second-order cone problem (MISOCP), which we can solve effectively with Mosek [7], Gurobi [6], and other tools.

D. Choosing an Objective Function

Mellinger et al. relate the snap (that is, the fourth derivative of position) to the control inputs of a quadrotor, and thus choose an objective of the form:

$$\text{minimize} \sum_{j=1}^{N} \int_0^1 \left\| \frac{d^4}{dt^4} P_j(t) \right\|^2 dt \tag{18}$$

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. We demonstrate this objective function in operation with 5$^{\text{th}}$-degree polynomials in Fig. 4.

However, to reduce our problem to a MISOCP and improve the numerical stability of the solver, we found it beneficial to restrict ourselves to 3$^{\text{rd}}$-degree polynomials and thus piecewise constant jerk. Our objective is

$$\text{minimize} \sum_{j=1}^{N} \left\| \frac{d^3}{dt^3} P_j(t) \right\|^2. \tag{19}$$

which is likewise convex and quadratic in the coefficients of the P_j. We also add linear constraints 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.

E. 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$^{\text{rd}}$ derivative. It thus has delta functions for its 4$^{\text{th}}$ derivative, which Mellinger relates directly to the rotor thrusts of the UAV [3]. Since this is clearly undesirable, we proceed as follows: First, we run the MISOCP to optimize our degree-3 polynomials and assign them to convex safe regions. Next, we fix the resulting assignment of trajectories to safe regions and then re-run 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. For example, in the office environment shown in Fig. 1, computing the smooth 5th-degree polynomial trajectory required only 1.5 s in Mosek.

F. Complete Formulation

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

$$\underset{P,H,\sigma}{\text{minimize}} \sum_{j=1}^{N} \left\| \frac{d^3}{dt^3} P_j(t) \right\|^2 \tag{20}$$

subject to:

$$\begin{aligned} &P_1(0) = X_0, && \dot{P}_1(0) = \dot{X}_0, && \ddot{P}_1(0) = \ddot{X}_0 \\ &P_N(1) = X_f, && \dot{P}_N(1) = \dot{X}_f, && \ddot{P}_N(1) = \ddot{X}_f \\ &P_j(1) = P_{j+1}(0), && \dot{P}_j(1) = \dot{P}_{j+1}(0), && \ddot{P}_j(1) = \ddot{P}_{j+1}(0) \end{aligned} \tag{21}$$

$$\begin{aligned} &H_{r,j} \implies b_{r,\ell} - a_{r,\ell}^\top P_j(j) = t\sigma_{\ell,j,1}(t) + (1-t)\sigma_{\ell,j,2}(t) \\ &\quad \forall j \in \{1,\dots,N\} \quad \forall r \in \{1,\dots,R\} \\ &\quad \text{where } \sigma_{\ell,j,1}(t),\ \sigma_{\ell,j,2}(t) \text{ are sums of squares} \end{aligned} \tag{22}$$

$$\sum_{r=1}^{R} H_{r,j} = 1 \quad \forall j \in \{1,\dots,N\} \tag{23}$$

$$H_{r,j} \in \{0,1\} \tag{24}$$

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.

G. Trajectories Without Convex Segmentation

For the sake of comparison, we have included numerical experiments for the proposed mixed-integer optimization using the faces of obstacles instead of convex regions from IRIS, just as Bellingham [16], Mellinger [9], and others have done. Instead of a single matrix of binary variables H, we have a matrix H_o for each obstacle. The m linear constraints in (5) are replaced with a linear constraint for each face r of obstacle o, and we constrain that

$$\sum_{r=1}^{N_{\text{faces}}} H_{o,r,j} = 1 \quad \forall j \in \{1,\dots,N\} \tag{25}$$

for every obstacle $o \in \{1,\dots,N_{\text{obstacles}}\}$. The disadvantage of this formulation is the rapid increase in the number of binary variables as the numbers of obstacles and faces increase. This increases the time required to find the global optimum, as shown in Fig. 6.

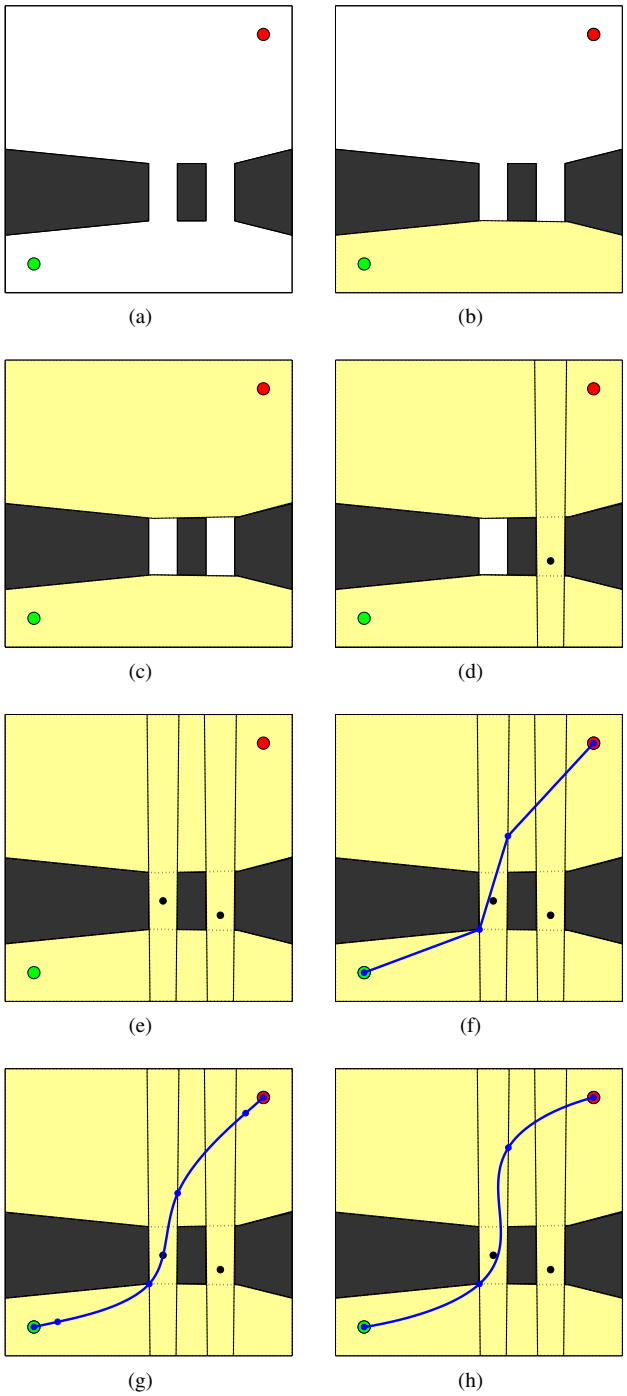


Fig. 4. Solving a simple environment with IRIS regions. We construct an environment with obstacles and a start and goal pose (a), then generate IRIS regions around the start (b) and goal (c). Next, we identify a point far from the existing set of obstacles and IRIS regions and seed a new region there (d), and repeat until we have 4 regions (e). Finally, we solve for trajectories of 1st-degree polynomials minimizing squared velocity in 0.1s (f), 3rd-degree polynomials minimizing squared jerk in 1.3s (g), and 5th-degree polynomials minimizing squared snap in 4.0s (h). All trajectories lie entirely within the convex regions shown.

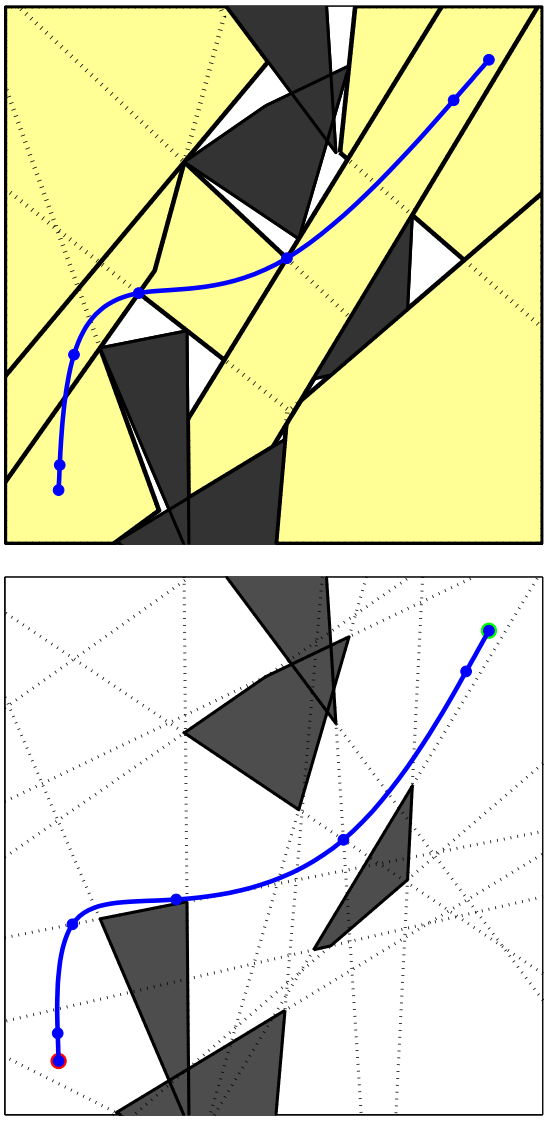


Fig. 5. An environment consisting of 5 uniformly randomly placed convex obstacles. Above: We generate 7 convex regions of free space using IRIS, then solve for a piecewise 3rd-degree trajectory which is entirely contained within those safe regions in 14.1s. Below: we also solve for a trajectory of the same degree without the convex segmentation step, which results in a 15% decrease in the optimal cost value but requires 27.5s to solve to optimality.

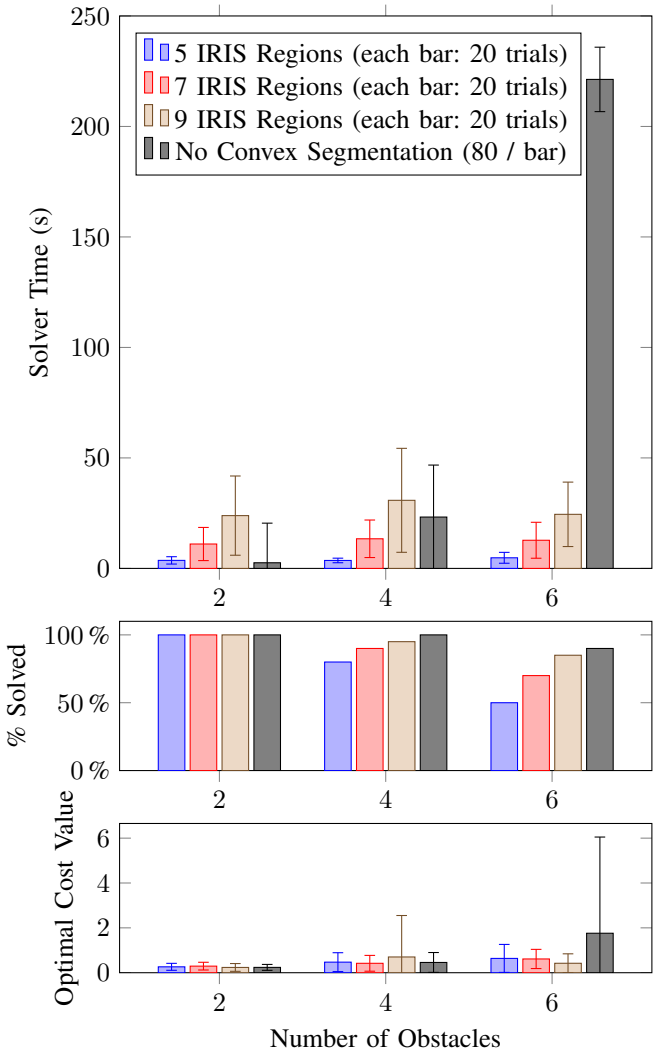


Fig. 6. Comparison of our approach for various numbers of convex safe regions, as well as the approach described in Sect. II-G, which does not require convex segmentation. We show results for randomly generated environments with 2, 4, or 6 obstacles. Above: the mean and std. dev. of time required to solve the problem to within 1% of global optimality using Mosek [7] on a 2.7 GHz 12-Core Intel Xeon E5 processor. For more than 4 obstacles, the solve time required increases dramatically if no convex segmentation is performed. Middle: the fraction of environments for which an optimal solution could be found. Reducing the number of IRIS regions improves the speed of optimization but, by covering less of the free space, it also decreases the likelihood of a feasible trajectory from start to goal being found. Below: the final value of the objective function at optimality in each case.

III. RESULTS

We demonstrate the mixed-integer trajectory planning in a variety of two- and three-dimensional environments. Figure 4 shows a simple 2D environment, in which we find four convex safe regions with IRIS and then solve for several trajectories through those regions. The arrangement of the obstacles in the simple environment is such that only four convex regions are needed to completely fill the space. For more complex environments, such as that shown in Fig. 5, more convex regions may be required, and those regions may not cover the entire space.

In Fig. 5, we show a randomly generated environment with five obstacles, a starting pose in the upper right, and a goal pose in the lower left. We generate 7 convex safe regions with IRIS, which do not entirely fill the obstacle-free space. Within those 7 regions, we plan 6 polynomial segments of degree 3 to form a smooth trajectory from the start to the goal while minimizing the objective in (19). This trajectory is shown in Fig. 5a. We can also avoid pre-computing convex regions and use the faces of the obstacles directly, as shown in Fig. 5b. This results in an optimal solution with a 15% lower (i.e. better) objective function value, but requires nearly double the solver time for the example shown. The solutions in Figs. 5a and 5b are both

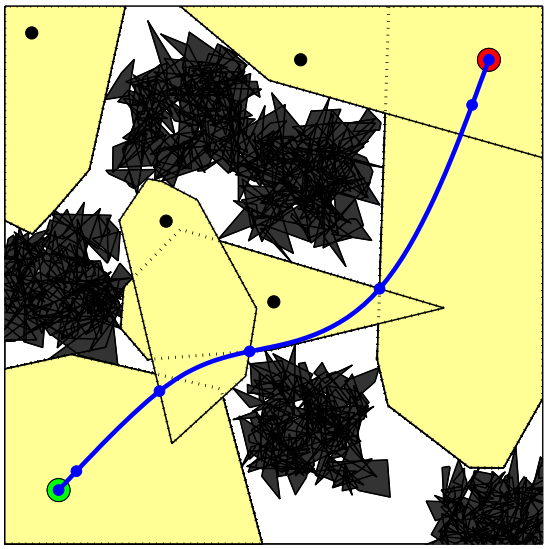


Fig. 7. Collision-free trajectory with very many obstacles. Five clusters of 100 obstacles each were placed, and 6 convex regions were found with IRIS in the free space. Solve time for this trajectory was 20s. Generating IRIS regions required $<$ 1s.

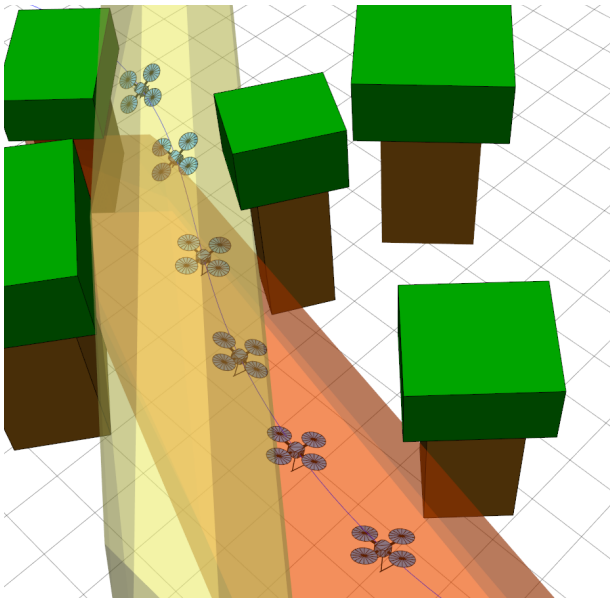


Fig. 8. A virtual forest environment showing two obstacle-free IRIS regions and a trajectory which passes through those regions. The UAV itself is treated as a sphere to ensure that no part of the vehicle is allowed to exit the set of obstacle-free regions.

globally optimal with respect to the cost function in (19), but they differ from one another because they are subject to different safe region constraints. In Fig. 5a, each polynomial must be contained entirely within one of the seven convex safe regions shown, while in Fig. 5b, each polynomial must lie on the outside of one face of every obstacle.

We compare the approach of generating convex safe regions (which may fail to fill the entire free space) with the approach of using the obstacle faces directly as our safe regions (which may require a dramatically more complex integer program) in Fig. 6. Environments consisting of 2, 4, or 6 obstacles were generated, and 5, 7, or 9 IRIS regions were automatically created using the heuristic described in Sect. I-A. Time spent on IRIS segmentation is not included in the table, but was less than 1 s in all cases. For each environment, we optimized a trajectory of 6 polynomial pieces of degree 3, while minimizing the objective in (19). We also attempted to find a trajectory in each environment using the method of Sect. II-G with no IRIS regions, which tended to be much slower for more than 4 obstacles. One such environment is shown in Fig. 5.

Substantially more complex environments can also be handled by the technique introduced here. In Fig. 7, we generate an environment of 500 obstacles in 5 clusters. Although we cannot in general hope to fully explore all possible paths through all 500 obstacles, our ability to quickly find large open areas of space with IRIS allows us to find a collision-free trajectory even in this extremely cluttered space.

We are by no means limited to problems in two dimensions. In Fig. 1, we show trajectories generated in two different 3D environments. For each environment, we generated 7 to 9 regions of safe space in 3D with IRIS, then planned a trajectory consisting of 7 degree-3 polynomials assigned to those safe regions. Each trajectory took approximately 200 s to solve to within 1% of the globally optimal objective value on an Intel i7 at 2.9GHz. The convex regions were generated for the configuration space of a bounding sphere for the UAV in order to ensure that the trajectory would be collision-free for the whole vehicle. Figure 8 shows two of the convex regions used in the forest environment, along with part of the trajectory through those regions.

IV. CONCLUSION

We present 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.

A. 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 the

environments shown in Figs. 4, 5, and 7 and in the forest environment shown in Fig. 1, 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 in this case: in the office environment shown in Fig. 1, 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 Yalmip and Mosek without encountering numerical difficulties. The choice of basis functions Φ in Eq. 3 is likely to be a significant factor in the numerical stability of the solver [9]. So far, we have experimented with the Legendre basis suggested by Mellinger et al., but not with other polynomial bases.

B. Future Work

In the future, we intend to explore additional constraints and objectives, such as waypoints in space which must be visited en route from the start to the goal. We also plan to 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 bring these trajectories into the real world with their stabilization and execution on hardware.

V. ACKNOWLEDGMENTS

The authors are grateful to all of the members of the Robot Locomotion Group at MIT, and particularly to Andres Valenzuela, Frank Permenter, Anirudha Majumdar, and Scott Viteri for their ideas, time, and expertise.

VI. SOURCE CODE

The MATLAB code used to generate the trajectories shown in this paper is included within the Drake toolbox, which is freely available at https://github.com/RobotLocomotion/drake.

REFERENCES

[1] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Workshop on the Algorithmic Foundations of Robotics*, Istanbul, Turkey, 2014. [Online]. Available: http://groups.csail.mit.edu/robotics-center/public_papers/Deits14.pdf

[2] R. Tedrake, "Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems," 2014. [Online]. Available: http://drake.mit.edu

[3] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 2520–2525.

[4] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge, UK; New York: Cambridge University Press, 2004.

[5] R. M. Karp, "Reducibility among combinatorial problems," in *50 Years of Integer Programming 1958-2008*, M. Jnger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, Eds. Springer Berlin Heidelberg, 2010, pp. 219–241. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-540-68279-0_8

[6] Gurobi Optimization, Inc., "Gurobi optimizer reference manual," 2014. [Online]. Available: http://www.gurobi.com/

[7] Mosek ApS, "The MOSEK optimization software," 2014. [Online]. Available: http://www.mosek.com/

[8] IBM Corp., "User's manual for CPLEX," 2010. [Online]. Available: http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r2/topic/com.ibm.common.doc/doc/banner.htm

[9] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, May 2012, pp. 477–483.

[10] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *European Control Conference*, Porto, Portugal, 2001.

[11] A. Richards, J. Bellingham, M. Tillerson, and J. How, "Coordination and control of multiple UAVs," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*. Monterey, CA: American Institute of Aeronautics and Astronautics, Aug. 2002. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/6.2002-4588

[12] K. F. Culligan, "Online trajectory planning for UAVs using mixed integer linear programming," Thesis, Massachusetts Institute of Technology, 2006, thesis (S.M.)–Massachusetts Institute of Technology, Dept. of Aeronautics and Astronautics, 2006. [Online]. Available: http://dspace.mit.edu/handle/1721.1/37952

[13] Y. Hao, A. Davari, and A. Manesh, "Differential flatness-based trajectory planning for multiple unmanned aerial vehicles using mixed-integer linear programming," in *PROCEEDINGS OF THE AMERICAN CONTROL CONFERENCE*, vol. 1, 2005, p. 104. [Online]. Available: http://www.ece.arizona.edu/~sprinkjm/research/c2wt/uploads/Main/paper4.pdf

[14] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," *IEEE-RAS International Conference on Humanoid Robots*, Nov. 2014. [Online]. Available: http://groups.csail.mit.edu/robotics-center/public_papers/Deits14a.pdf

[15] A. Richards and J. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 3, 2002, pp. 1936–1941 vol.3.

[16] J. S. Bellingham, "Coordination and control of uav fleets using mixed-integer linear programming," Ph.D. dissertation, Citeseer, 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.2555&rep=rep1&type=pdf

[17] M. E. Flores, "Real-time trajectory generation for constrained nonlinear dynamical systems using non-uniform rational b-spline basis functions," Ph.D. dissertation, California Institue of Technology, Pasadena, CA, 2007.

[18] J. Lofberg, "Big-m and convex hulls," 2012. [Online]. Available: http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Tutorials.Big-MAndConvexHulls

[19] P. A. Parrilo, "Sums of squares and semidefinite programming," Mar. 2006. [Online]. Available: http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-972-algebraic-techniques-and-semidefinite-optimization-spring-2006/lecture-notes/lecture_10.pdf

[20] V. Powers and T. Wrmann, "An algorithm for sums of squares of real polynomials," *Journal of Pure and Applied Algebra*, vol. 127, no. 1, pp. 99–104, May 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022404997838273

[21] J. Lofberg, "YALMIP wiki," 2012. [Online]. Available: http://users.isy.liu.se/johanl/yalmip/

[22] S. Mars and L. Schewe, "An SDP-package for SCIP," Technical report, TU Darmstadt, Tech. Rep., 2012. [Online]. Available: http://www.opt.tu-darmstadt.de/~smars/SDP/scip_sdp_preprint.pdf