BadgerWorks

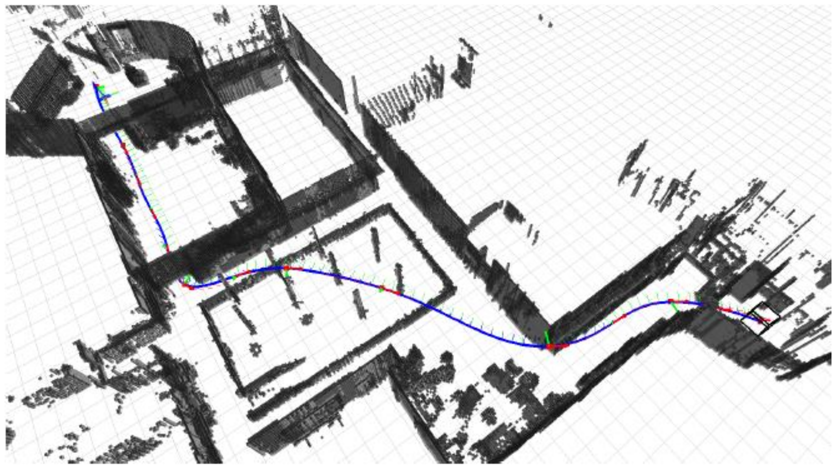
Topic: Polynomial Trajectory Planning for Quadrotor Flight

Dr. Kostas Alexis (CSE)

Slides based on the paper of C. Richter, A. Bry, and N. Roy

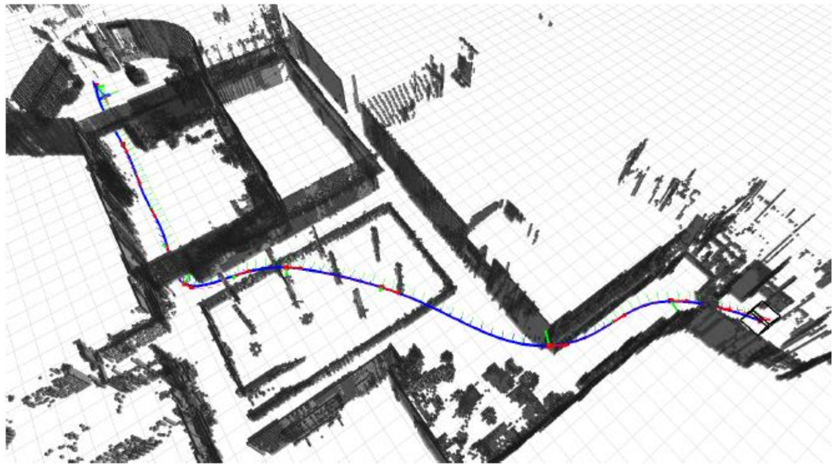
Motivation

- The goal is to enable planning for collision free navigation through complex environments for quadrotors (and broadly what we call Multirotor Aerial Vehicles).
- The method jointly optimizes polynomial path segments and exploits the differential flatness of quadrotors.



Motivation

- The goal of this work is to simultaneously find and optimize a collision free quadrotor trajectory through a complex real-world environments in a manner that is computationally efficient for real-time operation.
- Explicit optimization step for high-speed trajectories.



Problem Statement

- Given a 3D occupancy map of an environment, the goal is to efficiently compute:
  - Feasible
  - Minimum-snap

Trajectories that follow the shortest route from start to goal utilizing the full dynamic capabilities of the quadrotor

Solution Outline

- Argument: while it is possible to compute quadrotor trajectories using existing methods for sampling-based kinodynamic planning or trajectory optimization, these methods can be inefficient or impractical in complex real-world environments.
- The proposed solution is to utilize RRT* to find a route through the environment, initially ignoring the dynamics of the vehicle. That route is pruned to a sequence of waypoints representing the optimal route through the visibility graph of the environment. Then a sequence of polynomial segments is jointly optimized to join those waypoints into a smooth minimum-snap trajectory from start to goal. The method utilizes a differentially flat model of the quadrotor and the associated control techniques.

Quadrotor Dynamics and Control

- In order to guarantee that we can precisely follow the polynomial trajectories to be generated, the differential flatness property is utilized for the quadrotor equations of motion:

$$
\begin{aligned}
m\ddot{\mathbf{r}} &= mg\mathbf{z}_W - f\mathbf{z}_B \\
\dot{\omega} &= J^{-1}\left[-\omega \times J\omega + M\right]
\end{aligned}
$$

- A polynomial trajectory segment is in fact three polynomial functions of time specifying the independent evolution of the so-called flat output variables, x,y, and z between two positions in 3D space.
- The nonlinear controller employed to follow differential trajectories (may) take the form:

$$
\begin{aligned}
f &= \left(-k_x e_x - k_v e_v + mg\mathbf{z}_W + m\ddot{\mathbf{r}}_d\right) \cdot R\mathbf{z}_w \\
M &= -k_R e_R - k_\omega e_\omega + \omega \times J\omega \\
&\quad - J\left(\hat{\omega} R^T R_d \omega_d - R^T R_d \dot{\omega}_d\right)
\end{aligned}
$$

Quadrotor Dynamics and Control

- The intuition behind differential flatness lies in the fact that the quadrotor must always align its axis of thrust with the total acceleration vector prescribed at every point along the trajectory, thus determining its exact orientation and required control inputs.
- If the model equations were a perfect representation of the dynamics, and in the absence of disturbances, the feed-forward term would carry the quadrotor precisely across the trajectory.
- Since the desired trajectory and its derivatives are sufficient to compute the states and control inputs at every point along the path in closed form, it effectively serves as a simulation of the vehicle's motion.

Polynomial Trajectory Optimization

- Goal: analytical method for generating minimum-snap polynomial trajectories to be followed by a quadrotor that uses the control techniques described before.
- It is assumed that we have obtained a sequence of waypoints in 3D space representing the optimal path through the visibility graph of the environment (e.g., from RRT*) and we wish to generate a minimum-snap polynomial path passing through each of these waypoints.
- The choice of polynomial trajectories is natural for highly dynamic vehicles since these trajectories can be obtained efficiently as the solution to a quadratic program (QP) that minimizes a cost function of the path derivatives.
  - This optimization framework allows the endpoints of the path segments to be optimally fixed to desired values or left free, and the polynomials can be jointly optimized while maintaining continuity of the derivatives up to an arbitrary selected order.
- First a constrained QP formulation is derived and then reformulated as an unconstrained one (for numerical stability).

Cost Function for Minimizing Derivatives

- For quadrotors, a single trajectory segment between two points is composed of independent polynomial trajectories for the flat output variables x,y,z,ψ. Each polynomial segment takes the form:

$$P(t) = p_n t^N + P_{n-1} t^{N-1} + \cdots + p_0 = \sum_{n=0}^{N} p_n t^n$$

- The cost function for optimization of each polynomial is:

$$J = \int_0^T c_0 P(t)^2 + c_1 P'(t)^2 + c_2 P''(t)^2 + \ldots + c_N P^{(N)}(t)^2 dt$$

- Where T is the traversal time for the trajectory segment.

Cost Function for Minimizing Derivatives

- To solve for a minimum-snap trajectory, c_4 would be nonzero while all the other coefficients would be set to zero. This function can be written in matrix form:

$$J = \bar{p}^T Q \bar{p}$$

- Where $\bar{p}$ is a vector of polynomial coefficients and Q is a cost matrix constructed as the weighted sum of Hessian matrices for each of the polynomial derivatives.
  - Hessian matrices will be derived by writing the cost function in terms of the polynomial coefficients and then differentiating twice with respect to those coefficients.

Cost Function for Minimizing Derivatives

- To solve for a minimum-snap trajectory, c_4 would be nonzero while all the other coefficients would be set to zero. This function can be written in matrix form:

$$J = \bar{p}^T Q \bar{p}$$

- Where $\bar{p}$ is a vector of polynomial coefficients and Q is a cost matrix constructed as the weighted sum of Hessian matrices for each of the polynomial derivatives.
  - Hessian matrices will be derived by writing the cost function in terms of the polynomial coefficients and then differentiating twice with respect to those coefficients.

Cost Function for Minimizing Derivatives

- To solve for a minimum-snap trajectory, c_4 would be nonzero while all the other coefficients would be set to zero. This function can be written in matrix form:

$$J = \bar{p}^T Q \bar{p}$$

- Where $\bar{p}$ is a vector of polynomial coefficients and Q is a cost matrix constructed as the weighted sum of Hessian matrices for each of the polynomial derivatives.
  - Hessian matrices will be derived by writing the cost function in terms of the polynomial coefficients and then differentiating twice with respect to those coefficients.
  - We begin by writing the square of the polynomial as a convolution sum:

$$(P^2)_n = \sum_{j=0}^{N} p_j p_{n-j}$$

  - Where $(P^2)_n$ is the n-th coefficient of the squared polynomial.

Cost Function for Minimizing Derivatives

- To solve for a minimum-snap trajectory, c_4 would be nonzero while all the other coefficients would be set to zero. This function can be written in matrix form:

$$J = \bar{p}^T Q \bar{p}$$

- Where $\bar{p}$ is a vector of polynomial coefficients and Q is a cost matrix constructed as the weighted sum of Hessian matrices for each of the polynomial derivatives.
  - Hessian matrices will be derived by writing the cost function in terms of the polynomial coefficients and then differentiating twice with respect to those coefficients.
  - The r-th derivative of a polynomial is

$$P^{(r)}(t) = \sum_{n=r}^{N} \left(\prod_{m=0}^{r-1} (n-m) \right) p_n t^{n-r}$$

Cost Function for Minimizing Derivatives

- To solve for a minimum-snap trajectory, c_4 would be nonzero while all the other coefficients would be set to zero. This function can be written in matrix form:

$$J = \bar{p}^T Q \bar{p}$$

- Where $\bar{p}$ is a vector of polynomial coefficients and Q is a cost matrix constructed as the weighted sum of Hessian matrices for each of the polynomial derivatives.
  - Hessian matrices will be derived by writing the cost function in terms of the polynomial coefficients and then differentiating twice with respect to those coefficients.
  - Hence the component of the cost function associated with the r-th derivative is:

$$\begin{aligned} J_r &= \int_0^T P^{(r)}(t)^2 dt \\ &= \sum_{n=0}^{2N} \sum_{j=0}^{N} \left(\prod_{m=0}^{r-1} (j-m)(n-j-m) \right) p_j p_{n-j} \frac{T^{n-2r+1}}{n-2r+1} \end{aligned}$$

Cost Function for Minimizing Derivatives

- To solve for a minimum-snap trajectory, c_4 would be nonzero while all the other coefficients would be set to zero. This function can be written in matrix form:

$$J = \bar{p}^T Q \bar{p}$$

- Where $\bar{p}$ is a vector of polynomial coefficients and Q is a cost matrix constructed as the weighted sum of Hessian matrices for each of the polynomial derivatives.
  - Hessian matrices will be derived by writing the cost function in terms of the polynomial coefficients and then differentiating twice with respect to those coefficients.
  - Computation of the Hessian begins by differentiating J_r

$$\begin{aligned}\frac{\partial J_r}{\partial p_i} =& \sum_{n=0}^{2N} \sum_{j=0}^{N} \left(\prod_{m=0}^{r-1} (j-m)(n-j-m) \right) \\ & \cdot \left(\frac{\partial p_j}{\partial p_i} p_{n-j} + \frac{\partial p_{n-j}}{\partial p_i} p_j \right) \frac{T^{n-2r+1}}{n-2r+1} \\ =& 2 \sum_{n=0}^{2N} \left(\prod_{m=0}^{r-1} (i-m)(n-i-m) \right) \\ & \cdot p_{n-i} \frac{T^{n-2r+1}}{n-2r+1}\end{aligned}$$

Cost Function for Minimizing Derivatives

- To solve for a minimum-snap trajectory, c_4 would be nonzero while all the other coefficients would be set to zero. This function can be written in matrix form:

$$J = \bar{p}^T Q \bar{p}$$

- Where $\bar{p}$ is a vector of polynomial coefficients and Q is a cost matrix constructed as the weighted sum of Hessian matrices for each of the polynomial derivatives.
  - Hessian matrices will be derived by writing the cost function in terms of the polynomial coefficients and then differentiating twice with respect to those coefficients.
  - Then differentiating again wrt each of the polynomials:

$$\begin{aligned}\frac{\partial^2 J_r}{\partial p_i \partial p_l} &= 2 \sum_{n=0}^{2N} \left(\prod_{m=0}^{r-1} (i-m)(n-i-m) \right) \\ &\cdot \frac{\partial p_{n-i}}{\partial p_l} \frac{T^{n-2r+1}}{n-2r+1} \\ &= 2 \left(\prod_{m=0}^{r-1} (i-m)(l-m) \right) \frac{T^{i+l-2r+1}}{i+l-2r+1}\end{aligned}$$

Cost Function for Minimizing Derivatives

- To solve for a minimum-snap trajectory, c_4 would be nonzero while all the other coefficients would be set to zero. This function can be written in matrix form:

$$J = \bar{p}^T Q \bar{p}$$

- Where $\bar{p}$ is a vector of polynomial coefficients and Q is a cost matrix constructed as the weighted sum of Hessian matrices for each of the polynomial derivatives.
  - Hessian matrices will be derived by writing the cost function in terms of the polynomial coefficients and then differentiating twice with respect to those coefficients.
  - Finally

$$Q_r^{il} = \begin{cases} 2\left(\prod_{m=0}^{r-1}(i-m)(l-m)\right)\frac{T^{i+l-2r+1}}{i+l-2r+1} & \text{if } i \geq r \wedge l \geq r \\ 0 & \text{if } i < r \vee l < r \end{cases}$$

  - Where the complete cost matrix Q is given by:

$$Q = \sum_{r=0}^{N} c_r Q_r$$

  - With c_r being the user-specified penalty on the r-th derivative.

Constraints

- The constraints on the endpoints of a polynomial segment, which are used to either fix a given derivative to a desired value or ensure continuity of free derivatives, are imposed as a linear function of the coefficients:

$$A\bar{p} - b = 0$$

$$A = \begin{bmatrix} A_0 \\ A_T \end{bmatrix}, \quad b = \begin{bmatrix} b_0 \\ b_T \end{bmatrix}$$

- Where A is constructed by evaluating the component of the derivative in $P^{(r)}(t) = \sum_{n=r}^{N} \left(\prod_{m=0}^{r-1} (n-m) \right) p_n t^{n-r}$ corresponding to the appropriate coefficient

$$A_{0_{rn}} = \begin{cases} \prod_{m=0}^{r-1}(r-m) & \text{if } r = n \\ 0 & \text{if } r \neq n \end{cases}$$

$$b_{0_r} = P^{(r)}(0)$$

$$A_{T_{rn}} = \begin{cases} \left(\prod_{m=0}^{r-1}(r-m)\right) T^{n-r} & \text{if } n \geq r \\ 0 & \text{if } r < n \end{cases}$$

$$b_{T_r} = P^{(r)}(T)$$

Constraints

- These constraints either fix the position, velocity, acceleration, and higher order derivatives to desired values, or allow them to float subject to minimization of the cost function.
- Having assembled Q, A and b, the QP can be written as:

$$\min_{\bar{p}} \quad \bar{p}^T Q \bar{p}$$

$$\text{s.t.} \quad A\bar{p} - b = 0$$

- Where the decision variables are the coefficients of the polynomial trajectory.

Reformulation as Unconstrained QP

- The above method works well for single segments and small optimization problems but may become ill-conditioned for larger joint optimization problems.
- To avoid ill-conditioning, we reformulate the problem as an unconstrained QP to solve for endpoint derivatives directly as the decision variables, rather than the indirect method of solving for polynomial coefficients.
- Once the optimal waypoint derivatives are found, the minimum order polynomial connecting each pair of waypoints can be obtained by inverting the appropriate constraint matrix.

Reformulation as Unconstrained QP

- We begin with the original cost function

$$J = \bar{p}^T Q \bar{p}$$

- We utilize the A matrix as a mapping between polynomial coefficients and the endpoint derivatives

$$\bar{d} = A\bar{p} = \begin{bmatrix} A_0 \\ A_T \end{bmatrix} \bar{p}$$

- And therefore the cost function for a single polynomial segment takes the form

$$J = \bar{d}^T A^{-T} Q A^{-1} \bar{d}$$

Reformulation as Unconstrained QP

- We begin with the original cost function

$$J = \bar{p}^T Q \bar{p}$$

- We utilize the A matrix as a mapping between polynomial coefficients and the endpoint derivatives

$$\bar{d} = A\bar{p} = \begin{bmatrix} A_0 \\ A_T \end{bmatrix} \bar{p}$$

- And therefore the cost function for a single polynomial segment takes the form

$$J = \bar{d}^T A^{-T} Q A^{-1} \bar{d}$$

- **Goal (reminder!):** given an initial and a final state, and a sequence of intermediate waypoint locations, we wish to find the waypoint velocities, accelerations, and higher order derivatives such that the minimum-order polynomials connecting those waypoints wil minimize the aforementioned cost function.

Reformulation as Unconstrained QP

- In the case of a joint optimization, we construct a Q_{joint} and A_{joint} matrices, which are simply block diagonal matrices composed of the Q and A matrices for the individual subproblems.
- The derivatives involved in the joint optimization are concatenated into a vector D which typically includes a full set of derivatives to be fixed at the beginning and end of the trajectory (i.e., beging and end with zero velocity etc) alongside with the derivatives to be fixed at the waypoints (i.e., positions).
  - This formulation is easily adapted to fix or float any of the derivatives.

Reformulation as Unconstrained QP

- We sort D into a block of derivatives to be fixed in the optimization D_F and a block of free derivatives we intend to optimize (D_P)

$$D = \begin{bmatrix} D_F \\ D_P \end{bmatrix}$$

- We then rely on a selector matrix M to map the derivatives in D to an arrangement that is consistent with the sequence of block-diagonal elements in A_{joint}. In particular, since each block-diagonal element of A_{joint} represents the optimization of a single segment, the M matrix also serves to duplicate each intermediate waypoint derivative value to appear both at the end of one segment and at the beginning of the subsequent segment, therefore maintaining continuity of the derivatives.

Reformulation as Unconstrained QP

- New Total Cost Function for the Joint Optimization

$$J = \begin{bmatrix} D_F \\ D_P \end{bmatrix}^T MA^{-T}QA^{-1}M^T \begin{bmatrix} D_F \\ D_P \end{bmatrix}$$

- With R denoting the new augmented cost matrix

$$R = MA^{-T}QA^{-1}M^T$$

  - Constraints are now embedded within the cost function therefore yielding an unconstrained optimization problem.

Reformulation as Unconstrained QP

- Partitioning R according to the number of fixed and free derivatives and then expanding the cost function

$$R = \begin{bmatrix} R_{FF} & R_{FP} \\ R_{PF} & R_{PP} \end{bmatrix}$$

$$J = D_F^T R_{FF} D_F + D_F^T R_{FP} D_P + D_P^T R_{PF} D_F + D_P^T R_{PP} D_P$$

- Where the first term is simply a fixed cost incurred by satisfying the fixed derivatives.
- Differentiating J wrt D_p and equating to zero yields the optimal values for the free derivatives

$$D_P^* = -R_{PP}^{-1} R_{FP}^T D_F$$

- The optimal waypoint derivatives imply the minimum order polynomial segments needed to construct the complete trajectory.

Time Allocation

- So far we have required a fixed time associated with each segment in the complete trajectory since these times factor into the construction of the cost matrix.
- These segment times act as constraints on the solution quality, but can be allowed to vary to improve the overall solution.
- We therefore choose initial segment times and then iteratively refine the times in order to obtain better paths wrt the cost function.

Time Allocation

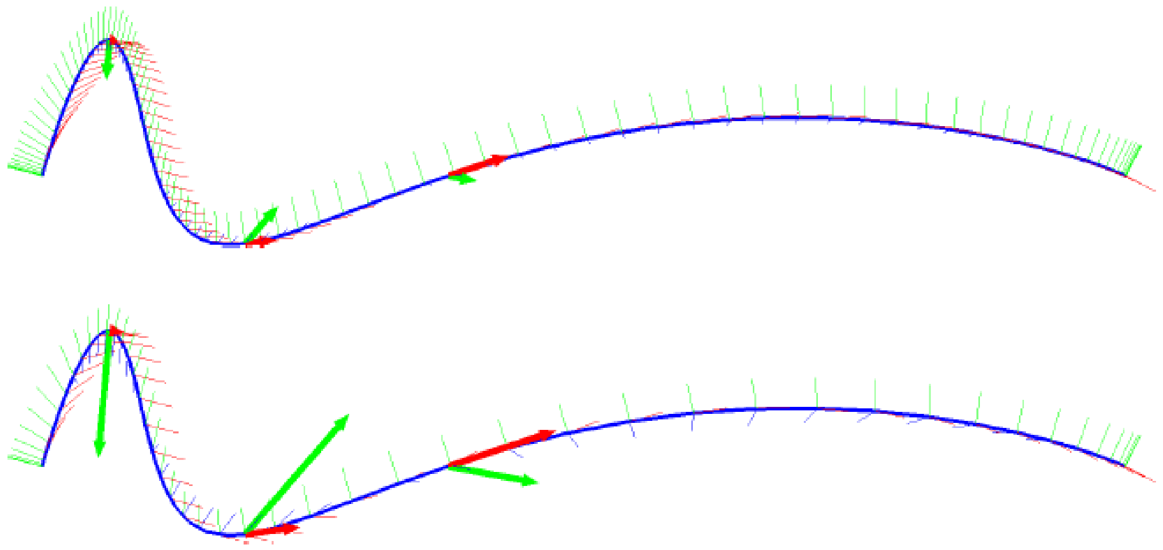
- Extension of the polynomial cost function to choose segment times and thus determine the total trajectory traversal time.
- We attempt to minimize

$$J_T = \bar{p}^T Q \bar{p} + k_T T$$

- Where T is the sum of segment times for the complete path and k_T is a user-specified penalty on time.
- The first time this cost function is simply the original cost function for polynomial optimization.
- When penalizing only acceleration, jerk or snap, this original cost can be driven arbitrarily small by increasing the total time.
  - This modified cost performs a trade off between minimizing the polynomial cost and traversing the path quickly.
  - Increasing the penalty on time, k_T, results in more aggressive maneuvers.

Time Allocation

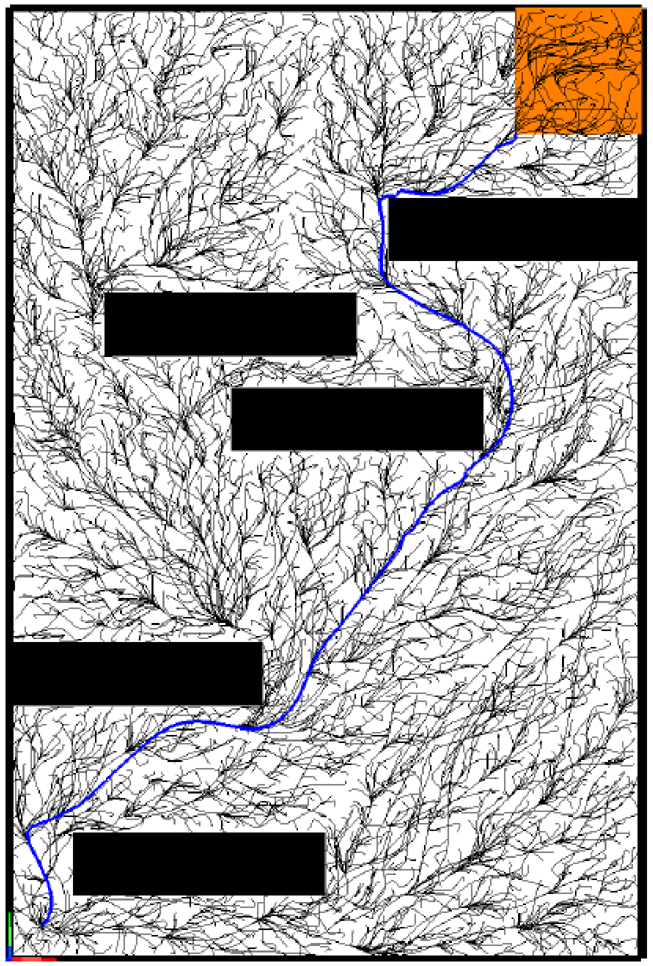
- We optimize the modified cost function via gradient descent
  - Gradient is estimated numerically by perturbing each segment by some time δt.
  - It is not required that the total path time is conserved in a given perturbation.
  - Therefore the path time will grow or diminish as necessary to optimize the modified cost.

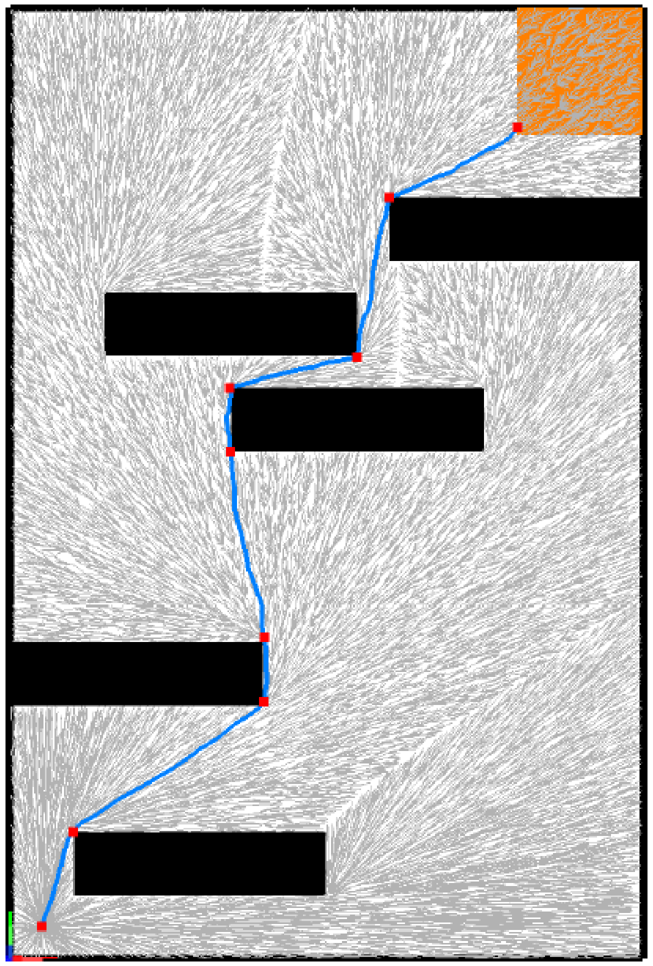


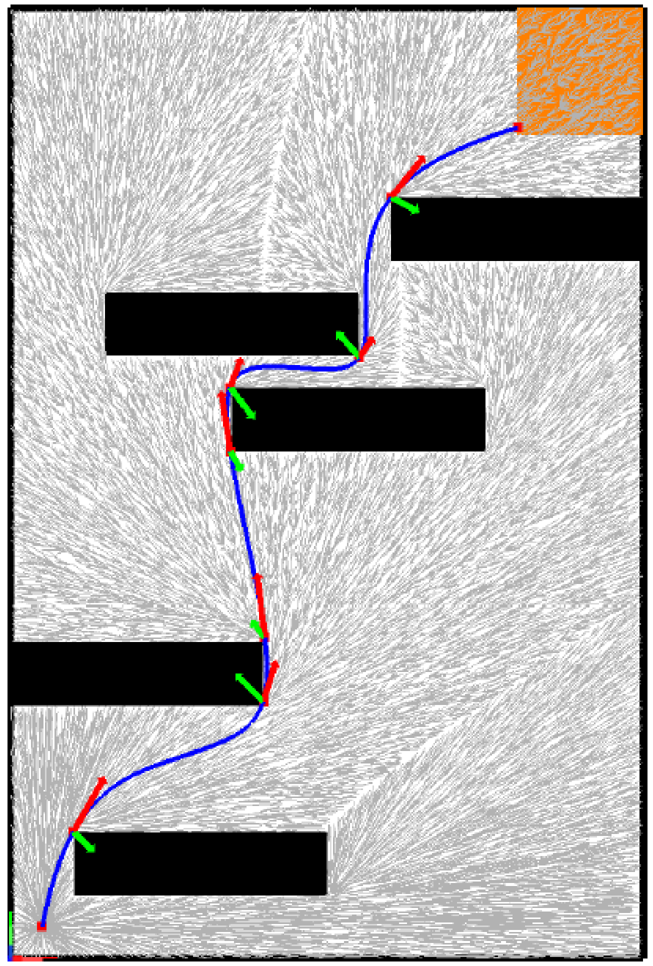
Ensuring Feasibility

- If a particular trajectory segment is found to intersect an obstacle after optimization, an additional waypoint is simply added halfway between its two ends, splitting this segment into two.
- This midpoint is known to be collision-free because it lies on the optimal route through the visibility graph.
- These additional waypoints are added incrementally until the polynomial trajectory is free of collision.

Indicative Result


(a) RRT* with polynomial steer function after 120s running time.


(b) Pruned waypoints from RRT* with straight-line steer function.


(c) Solution by our algorithm after 3s running time.

Find out more

- Richter, C., Bry, A. and Roy, N., 2016. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In Robotics Research (pp. 649-666). Springer, Cham.
- Charles, R., Bry, A. and Roy, N., 2013. Polynomial trajectory planning for quadrotor flight. In Proceedings of the IEEE International Conference on Robotics and Automation, ICRA.

Thank you!

Please ask your question!