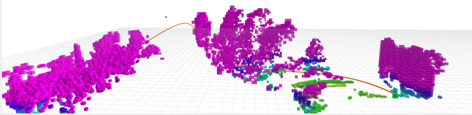
Motion Planning for Autonomous Vehicles

Gradient-based online trajectory generation

Geesara Kulathunga



April 28, 2023

Gradient-based online trajectory generation

Contents

The objective function

$$min \quad \lambda_1 f_s + \lambda_2 f_0 + \lambda_3 (f_v + f_a),$$

where f_s for cost for smoothness, f_o for cost clearance, f_a for cost for penalizing velocity and acceleration. Terms λ_1, λ_2, and λ_3 defined regularization terms.
For **defining polynomials** at each segment

$$\eta = M^{-1} C \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P^* \end{bmatrix}$$

where η defines the polynomial coefficients to **d**: $\mathbf{d_P}$ **free** derivatives and $\mathbf{d_F}$ **fixed** derivatives
Note: follow minimum-snap lecture materials for the derivation of this

Gao, F., Lin, Y., Shen, S. (2017, September). Gradient-based online safe trajectory generation for quadrotor flight in complex environments. In 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS) (pp. 3681-3688). IEEE.

The cost of the smoothness

$$
\begin{aligned}
f_s = \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix}^\top C M^{-\top} Q M^{-1} C^\top \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix} = \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix}^\top R \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix} \\
= \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix}^\top \begin{bmatrix} R_{FF} & R_{FP} \\ R_{PF} & R_{PP} \end{bmatrix} \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix} \\
= \mathbf{d}_F^\top R_{FF} \mathbf{d}_F + d_F^\top R_{FP} \mathbf{d}_P + \mathbf{d}_P^\top R_{PF} \mathbf{d}_F + \mathbf{d}_P^\top R_{PP} d_P \\
= \mathbf{d}_F^\top R_{FF} \mathbf{d}_F + 2\mathbf{d}_F^\top R_{FP} \mathbf{d}_P + \mathbf{d}_P^\top R_{PP} \mathbf{d}_P
\end{aligned} \tag{1}
$$

The Jacobian of f_s with respect to $\mathbf{d}_P$

$$J_s = \begin{bmatrix} \frac{\partial f_s}{\partial \mathbf{d}_{P_x}} & \frac{\partial f_s}{\partial \mathbf{d}_{P_y}} & \frac{\partial f_s}{\partial \mathbf{d}_{P_z}} \end{bmatrix}$$

where $\frac{\partial f_s}{\partial \mathbf{d}_{P_\mu}} = 2\mathbf{d}_F^\top R_{FP} + 2R_{PP}\mathbf{d}_P,\ \mu \in (x, y, z)$

The cost of the clearance A **differentiable function** to **penalize** the **distance** value, i.e., cost to rapidly grow up to infinity at where near the obstacles and to be flat at where away from the obstacles

$$c(d) = \alpha \cdot exp(-(d - d_0)/r),$$

where α is magnitude of the cost function, d_0 threshold value cost starts to rise, and r rate of the function rise

$$\begin{aligned} f_o &= \int_{T_0}^{T_M} c(p(t))ds \\ &= \int_{T_0}^{T_M} c(p(t)) \|v(t)\| \, dt = \Sigma_{k=0}^{\tau/\delta t} c(p(\tau_k)) \|v(t)\| \, \delta t, \end{aligned} \tag{2}$$

where $\tau_k = T_0 + k\delta t$, $v(t)$ is the velocity at position $p(t)$.

The Jacobian of f_o with respect to $\mathbf{d}_P$

$$\boldsymbol{J}_o = \begin{bmatrix} \frac{\partial f_o}{\partial \mathbf{d}_{P_x}} & \frac{\partial f_o}{\partial \mathbf{d}_{P_y}} & \frac{\partial f_o}{\partial \mathbf{d}_{P_z}} \end{bmatrix}$$

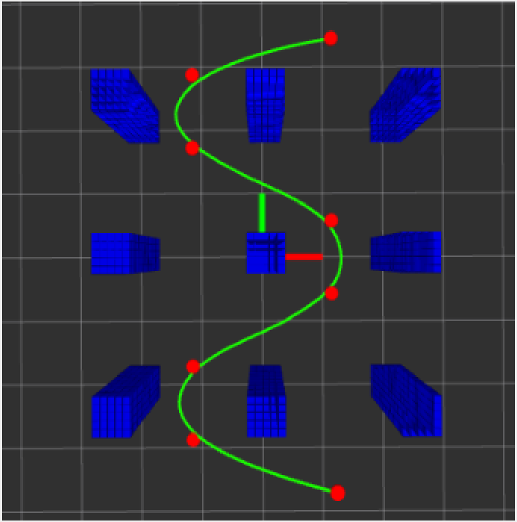
where $\frac{\partial f_s}{\partial \mathbf{d}_{P_\mu}} = \Sigma_{k=0}^{\tau/\delta t} \left\{ \nabla_\mu c(p(\tau_k)) \, \|v(t)\| \, \mathbf{F} + c(p(\tau_k)) \frac{v_\mu}{\|v(t)\|} \mathbf{G} \right\} \delta t$, $\mu \in (x, y, z)$

Let L_{dp} be right block of matrix $M^{-1}C$, i.e., free derivatives on the μ axis $d_{d\mu}$. Hence, $\mathbf{F} = TL_{dp}$, $\mathbf{G} = TV_m L_{dp}$, where V_m is the mapping matrix from the polynomial coefficients of the position to the polynomial coefficients of velocity, $T = [T_k^0, T_k^1, ..., T_k^n]$, and $\nabla_\mu c(p(\tau_k))$ is the gradient in the μ axis of the collision cost.

The cost of the dynamic feasibility An **artificial cost** field on **velocity** between the maximum velocity and minus maximum velocity

$$\begin{aligned} f_v =&\Sigma_{\mu \in \{x,y,z\}} \int_{T_0}^{T_M} c_v(v_\mu(t)) ds \\ =&\Sigma_{\mu \in \{x,y,z\}} \int_{T_0}^{T_M} c_v(v_\mu(t)) \left\| a(t) \right\| dt \end{aligned} \tag{3}$$

The Jacobian of J_v follows the similar formulation to J_o. Also, for finding f_a also follows same formulation as for f_v



An online replanner alongside with local planner (MPC)

Necessity of an online replanner

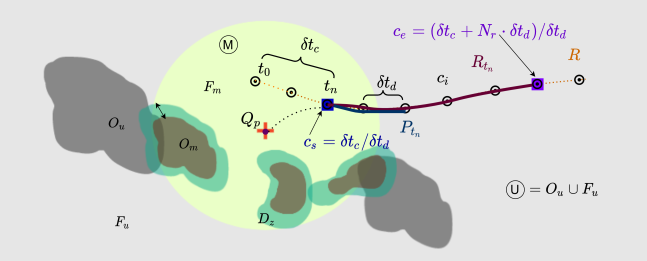
- Approximated motion model does not represent actual dynamics of the actual quadrotor
- **Local minima** difficult to maneuver through obstacle dense environments
- **Computational constraints** when increasing the **prediction horizon** of Model Predictive Control and the **number of obstacle constraints**

Kulathunga, G., Hamed, H., Devitt, D., Klimchik, A. (2022). Optimization-Based Trajectory Tracking Approach for Multi-Rotor Aerial Vehicles in Unknown Environments. IEEE Robotics and Automation Letters, 7(2), 4598-4605.

THE ONLINE REPLANNER

Problem Formulation

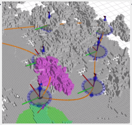
Develop a **soft constraint-based iteratively refine of the reference trajectory** to “push it out" of the obstacle-occupied space.



M $= O_m \cup F_m$ Map update range R_{t_n} Refining Horizon of Global Planner at t_n
R Initial Reference Trajectory P_{t_n} Prediction Horizon of Local Planner at t_n Q_p MAV pose

The online replanner

Objective function:
$J = \lambda_{smooth} J_{smooth} + \lambda_{obs} J_{obs} + \lambda_{feasibility} J_{feasibility}$, where $\lambda_*, * \in \{smooth, obs, feasibility\}$ are weight parameters are 0.2, 0.6, and 0.2

$T_{ref} = J\Big($  $\Big)$

$\frac{\partial(T_{ref})}{\partial c} = \frac{\partial J}{\partial c}\Big($  $\Big)$

$\frac{\partial J}{\partial c} = \lambda_{smooth} g_{smooth} + \lambda_{obs} g_{obs} + \lambda_{feasibility} g_{feasibility}$, where the control points of reference trajectory

Design considerations

- To improve the **smoothness**, part of the objective is dedicated to **minimize the acceleration and higher-order components**, e.g., snap, jerk, of the reference trajectory
- To solve the problem faster, designed as a **unconstrained optimization** problem, i.e., function minimizer. Thus, safety does not guarantee. Also, when refining reference trajectory, **feasibility constraints** on velocity and acceleration are enforced
- Solvers: LBFGS++ (`lbfgspp.statr.me`) and Mosek [2]

[2] Andersen, Erling D., and Knud D. Andersen. "The MOSEK interior-point optimizer for linear programming: an implementation of the homogeneous algorithm." High-performance optimization. Springer, Boston, MA, 2000. 197-232.

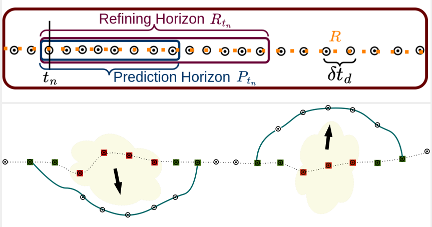
The high-level idea of reference trajectory tracker

Algorithm 1 Reference trajectory tracker

Inputs: at time t_n, R_{t_n}: reference trajectory to be refined, Q_p: current pose of MAV, P_{t_n}: trajectory to be tracked , M_{t_n}: EDT map of the environment
Outputs: R_{t_n}: refined reference trajectory, v_x, v_y, v_z, ω_z : control command to maneuver MAV

procedure GLOBAL PLANNER
  $R_{t_n} \leftarrow < Q_p, R_{t_n} >$
  $S_o \leftarrow$ CheckingOccupiedSegments(R_{t_n}, M_{t_n})
  if $S_o > 0$ **then**
    for $i \leftarrow S_o$ **do**
      $A_i, b_i \leftarrow$ ParallelConvexDecomposition(S_o^i)
      $S_*^i \leftarrow$ FindPushingDirections(S_o^i, A_i, b_i)
      $R_{t_n} \leftarrow$ CalculateGradients(S_*^i, R_{t_n})
  return $R_{t_n} \leftarrow$ ApplyBoxConstraintOptimization(R_{t_n})

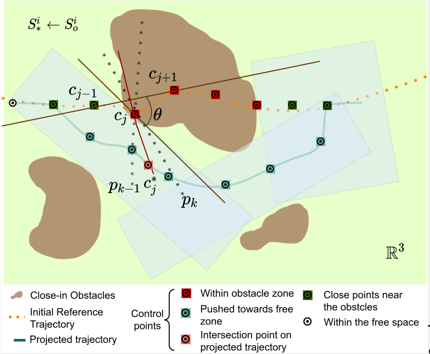
procedure LOCAL PLANNER
  $P_{t_n} \leftarrow < Q_p, P_{t_n} >$
  $C_o \leftarrow$ GetCloseInObstacles(P_{t_n}, M_{t_n})
  return $< v_x, v_y, v_z, \omega_z > \leftarrow$ ApplyNMPC(P_{t_n}, C_o)



Implemented the **parallel version** of convex decomposition [3]. Convex decomposition is applied to **successive control points** $CheckOccupiedSegments$, in parallel that result in the **free space H-rep** $Ax \leq b$ for each S_0^i

[3] S. Liu et al., "Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments," in IEEE Robotics and Automation Letters,

Finding Pushing Direction

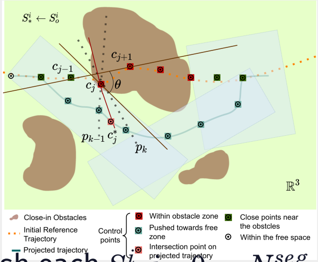


When **control points** in R_{t_n} can occur within the **obstacles zones**. Hence, control points that lie within the obstacle zone O_m must be pushed towards an obstacle-free zone

$$\min_{\mathbf{p}_0,\ldots,\mathbf{p}_n} \lambda_1 t_1 + \lambda_2 t_2 + t_3$$
$$\text{s.t.} \quad A\mathbf{p}_j \leq b,$$
$$\|\mathbf{p}_0 - \mathbf{c}_0\|_2 \leq t_1,$$
$$\|\mathbf{p}_n - \mathbf{c}_n\|_2 \leq t_2,$$
$$\sum_{j=1}^{n-1} \|\mathbf{p}_{j+1} - \mathbf{p}_j\|_2 \leq t_3,$$

A and b represents **the free space as a convex polyhedron** from $\mathbf{c}_0$ to $\mathbf{c}_n$ in S^i_o. $\lambda_1 = 0.8, \lambda_2 = 0.6$, and $\lambda_3 = 0.8$, were set in a way to provide more bias on **start** and **end** control points compared to **middle** control points

Gradient Direction Estimation



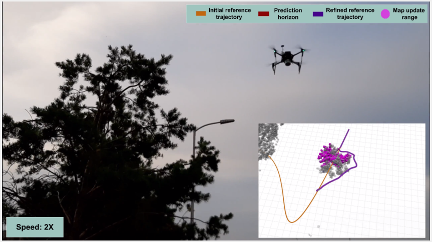
Push each $S^i_o, i = 0, ..., N^{seg}$ segment towards the obstacle-free zone N^{seg} is the number of segments that are within the obstacle zone for the considered refine trajectory segment R_{t_n}, at t

1. $\mathbf{v}_1 = \mathbf{c}_{j+1} - \mathbf{c}_{j-1}$ be the approximated direction vector along $\mathbf{c}_j$
2. $\mathbf{p}_k$ be the control point that intersects $\mathbf{v}_1$
3. corresponding direction vector $\mathbf{v}_2$ can be defined as $\mathbf{p}_k - \mathbf{c}_j$
4. $\theta = cos^{-1}(\mathbf{v}_1 \cdot \mathbf{v}_2 / \|\mathbf{v}_1\|_2 \|\mathbf{v}_2\|_2)$ between $\mathbf{v}_1$ and $\mathbf{v}_2$
5. $\mathbf{c}^{grad}_j = (\mathbf{c}^*_j - \mathbf{c}_j) / \left\| \mathbf{c}^*_j - \mathbf{c}_j \right\|_2$
6. $\mathbf{c}^*_j = \mathbf{p}_k + \frac{(\mathbf{p}_k - \mathbf{p}_{k-1})(\mathbf{v}_1 \cdot (\mathbf{c}_j - \mathbf{p}_k))}{\mathbf{v}_1 \cdot (\mathbf{p}_k - \mathbf{p}_{k-1})}$

$$J_{obs} = \Sigma_{i=d}^{N_r-d} J_{obs_i},$$

$$J_{obs_i} = \mathbf{v}_i \cdot dis_e^3, \quad \frac{\partial J_{obs_i}}{\partial \mathbf{c}_i} = -3 \cdot dis_e^2 \cdot \mathbf{c}_i^{grad},$$

where $dis_e = D_z - (\mathbf{c}_i - \mathbf{c}_i^*) \cdot \mathbf{c}_i^{grad}$ and $\mathbf{v}_i = \mathbf{c}_{i+1} - \mathbf{c}_i$, and avoidance distance D_z was set to 0.8m (distance must be higher than the radius of the MAV)



- **Smoothness** helps to reduce the effects such as **vibrations** caused due to higher-order components, i.e, jerk.
- The **velocity controller** is used in the proposed approach. Thus, higher-order components, e.g., **acceleration, jerk, snap**, should be minimized
- Minimizing the acceleration components

$$J_{smooth_i} = \mathbf{a}_i^\top \mathbf{a}_i, \quad \frac{\partial J_{smooth_i}}{\partial \mathbf{c}_i} = 2\frac{\partial \mathbf{a}_i}{\partial \mathbf{c}_i},$$

  where $\partial \mathbf{a}_i / \partial \mathbf{c}_i = 1$, $\partial \mathbf{a}_i / \partial \mathbf{c}_{i+1} = -2$, $\partial \mathbf{a}_i / \partial \mathbf{c}_{i+2} = 1$. $\mathbf{a}_i, \mathbf{v}_i, \mathbf{c}_i \in \mathbb{R}^3$ are respectively acceleration ($\mathbf{a}_i = \mathbf{c}_{i+2} - 2\mathbf{c}_{i+1} + \mathbf{c}_i$), velocity ($\mathbf{v}_i = \mathbf{c}_{i+1} - \mathbf{c}_i$), and control point at i^{th} index of R_{t_n}
- Adding **jerk** did not affect J_{smooth} considerably. Hence, only acceleration components were considered

Global Trajectory Refinement: Feasibility

To ensure the generated trajectory is **dynamically feasible** for the maneuver, **refined reference trajectory** is **bounded** to **velocity and acceleration** limits

$$
\begin{aligned}
J_{feasibility_i} &= (\mathbf{v}_i \oplus \mathbf{v}_{max})^\top (\mathbf{v}_i \oplus \mathbf{v}_{max}) \cdot \frac{1}{\delta^2} \\
&+ (\mathbf{a}_i \oplus \mathbf{a}_{max})^\top (\mathbf{a}_i \oplus \mathbf{a}_{max}) \\
\frac{\partial J_{feasibility_i}}{\partial \mathbf{c}_i} &= -2 \frac{\mathbf{v}_i \oplus \mathbf{v}_{max}}{\delta} \cdot \frac{1}{\delta^2} + 2 \frac{\mathbf{a}_i \oplus \mathbf{a}_{max}}{} \cdot \frac{1}{\delta^2},
\end{aligned}
$$

where the operator $\oplus$ is defined as

$$
\oplus = \begin{cases} - & if\ \mathbf{v}_i > \mathbf{v}_{max} \,||\, \mathbf{a}_i > \mathbf{a}_{max} \\ + & if\ \mathbf{v}_i < -\mathbf{v}_{max} \,||\, \mathbf{a}_i < -\mathbf{a}_{max} \\ not\ considering & otherwise \end{cases},
$$

where allowed maximum velocity and acceleration components are given by $\mathbf{v}_{max} \in \mathbb{R}^3$ and $\mathbf{a}_{max} \in \mathbb{R}^3$

Dead Zone Recovery

The **map construction** is **not precise** when the depth sensor has a **small FoV**. Also, **EDTM** building takes a considerable amount of **time** when the **environment is cluttered**. Therefore, the **local planner** may generate **control commands** that lead to quadrotor maneuvers into the D_z zone

$$\min_{\mathbf{q}_1,\ldots,\mathbf{q}_{N_r}} \sum_{l=1}^{N_r} q_l$$
$$\text{s.t.} \quad A(\mathbf{p}_l + \mathbf{c}_l) \le b,\ \|\mathbf{c}_l\|_2 \le q_l,\ l = 1,\ldots,N_r,$$

Such recovered control points are determined by $\mathbf{p}_l + \mathbf{c}_l$, where the control points $c_l, l = 1,\ldots,N_r$ are pushed and N_r number of control points in R_t