Real-Time Visual-Inertial Mapping, Re-localization and Planning Onboard MAVs in Unknown Environments

Seungwon Song
KAIST Robotics Program

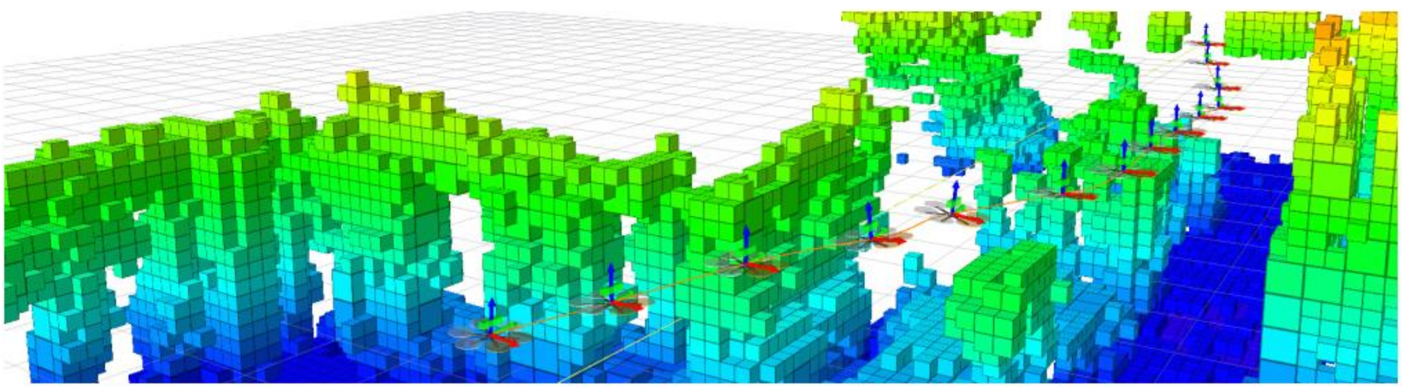
PURPOSE OF PAPER

Using

- Vision measurement (Visual Odometry)
- IMU measurement

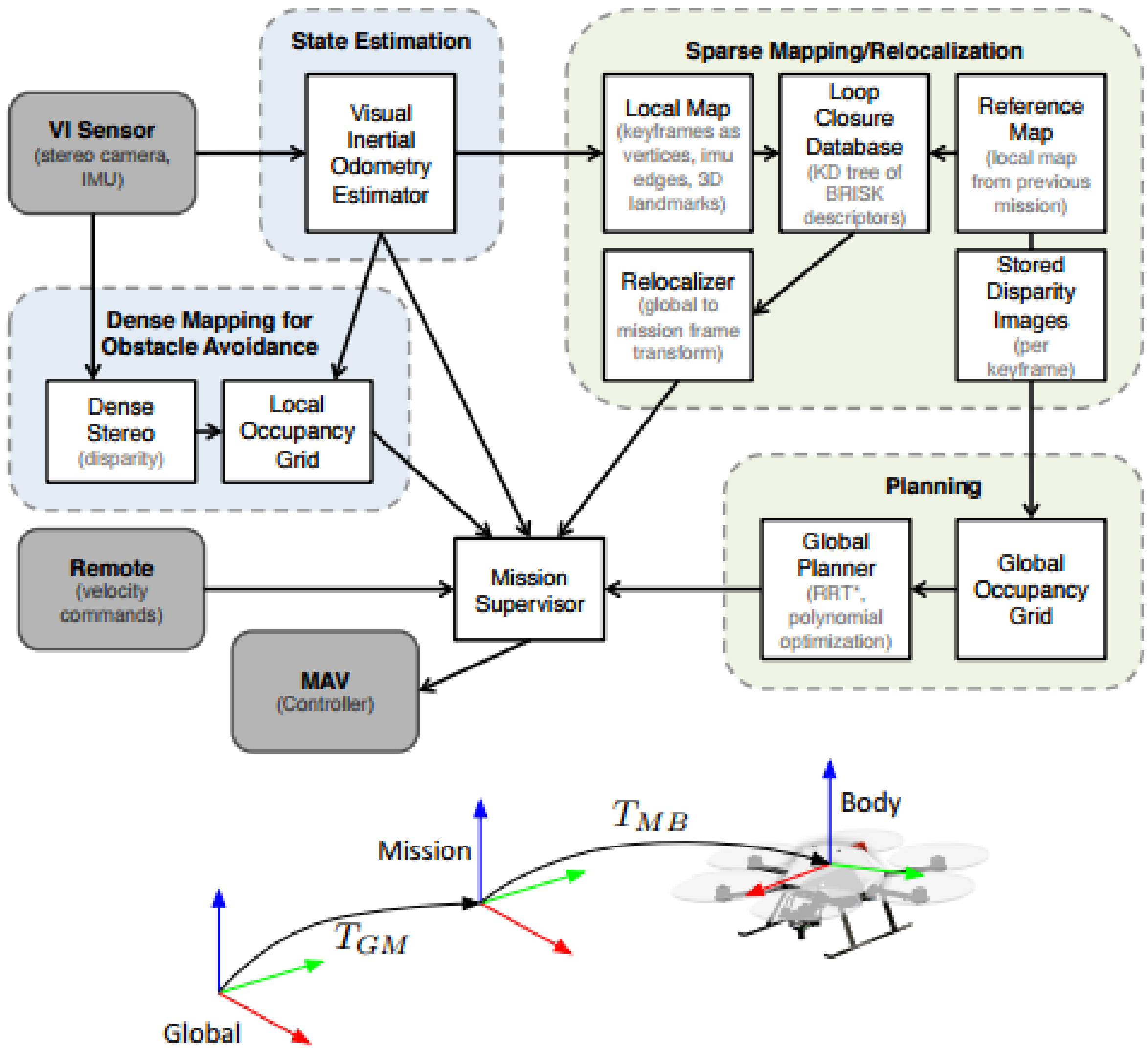
Purpose

- Create consistent maps
- To Quadrotor relocalize itself
- Plan path in full 3D

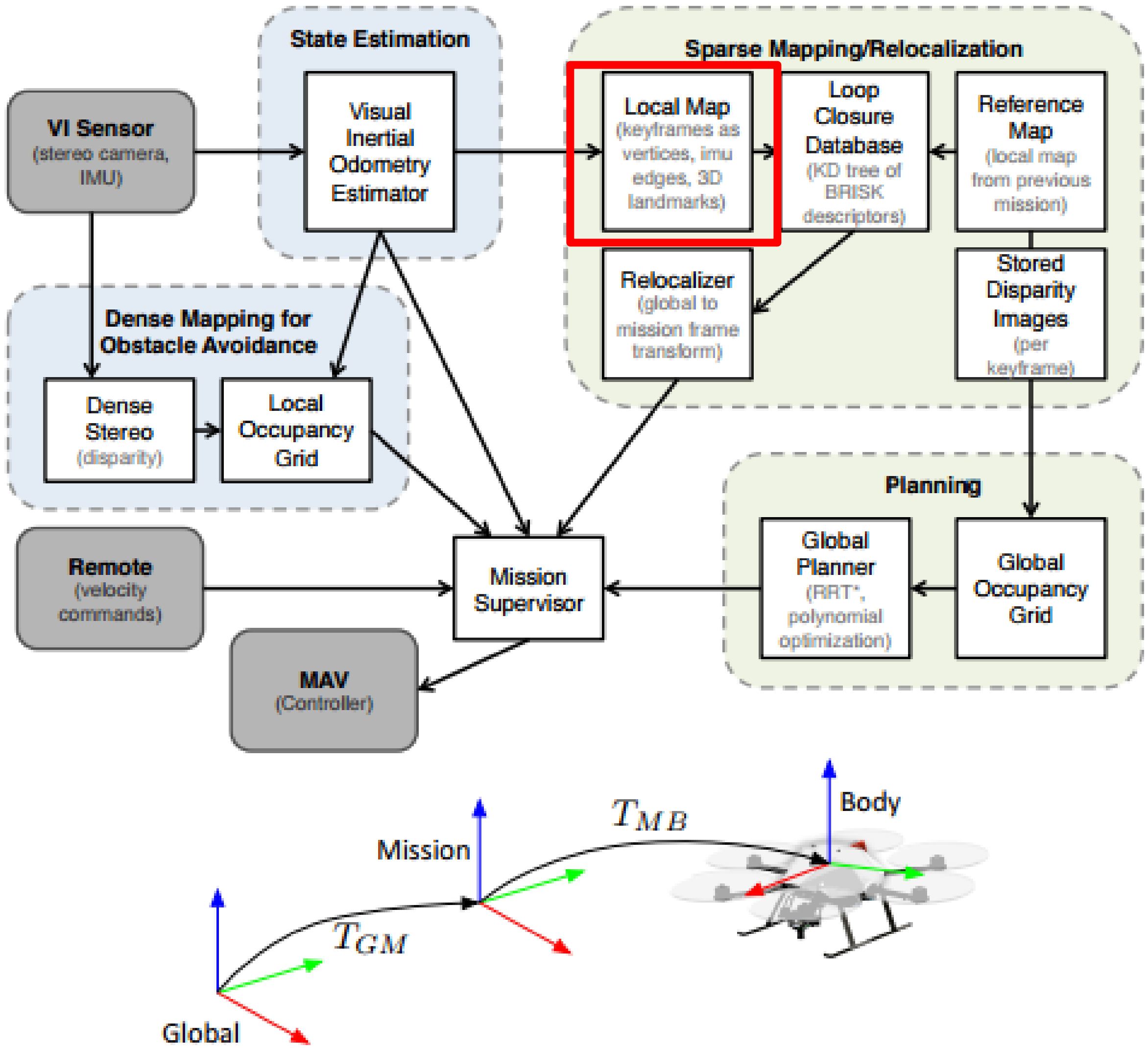


BASIC CONCEPT

- Local Map building
- Mission Handling
- Relocalization
- Pathplanning



LOCAL MAP BUILDING



LOCAL MAP BUILDING

- Normal Image Keypoint

Too much computation cost

⇓

- Make sparse pose-graph

-Vertices

Image keypoints

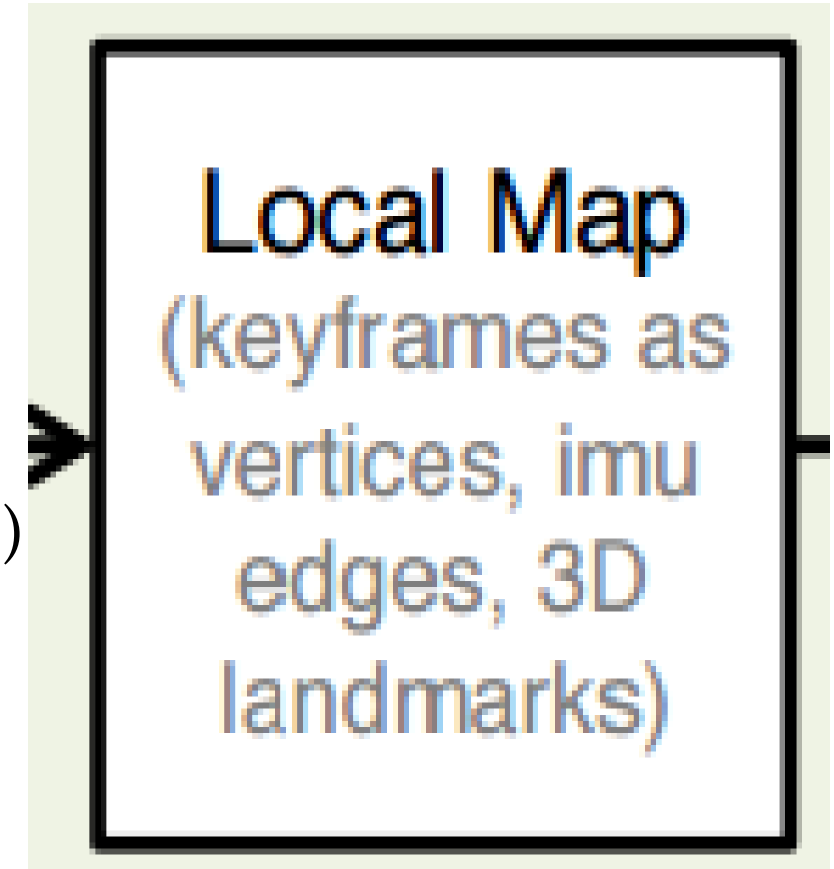
Keypoint descriptor

3D triangulated landmarks (stereo cam)

-Edge

IMU measurements

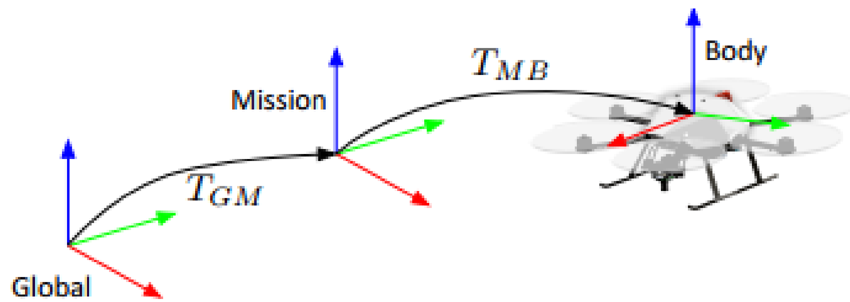
- Make Submissions with independent baseframes!



S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," The International Journal of Robotics Research, 2014

LOCAL MAP BUILDING

- Submission?



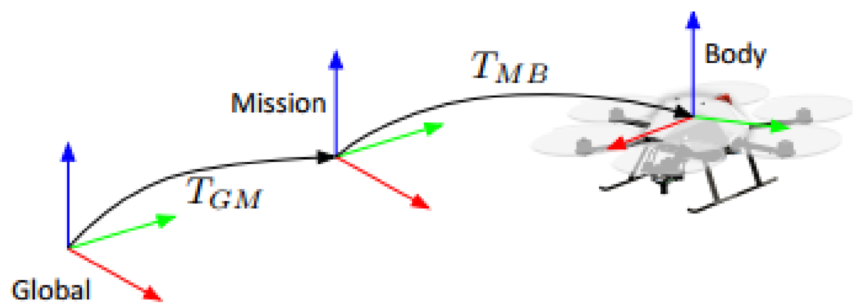
Reference Map

Translate Matrix	Translate Matrix	Translate Matrix	Translate Matrix
Local Map	Local Map	Local Map	Local Map

Mission Handling

- Add new local mission to Reference Map
- Do Bundle Adjustment (Reduce errors and drift in Odometry)
- Correct Translation ($T_{GM(original)} \rightarrow T_{GM(new)}$)

- Using Framework that allows to access the map from several threads. (Transaction change process + Incremental Local map)



T. Cieslewski, S. Lynen, M. Dymczyk, S. Magnenat, and R. Siegwart, “Map api - scalable decentralized map building for robots,” in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2015

Relocalization

- I think it's not so significant

- Matching BRISK keypoint descriptor
- Outlier rejection using RANSAC
- Add abstracted inliers as constraints
- Relocalize with past sliding windows
- Relocalize with low rate (estimator drift is slow relative to motion)

Pathplanning

- Based on RRT*
- Consider Cost!

  Human : Minimize the jerk !

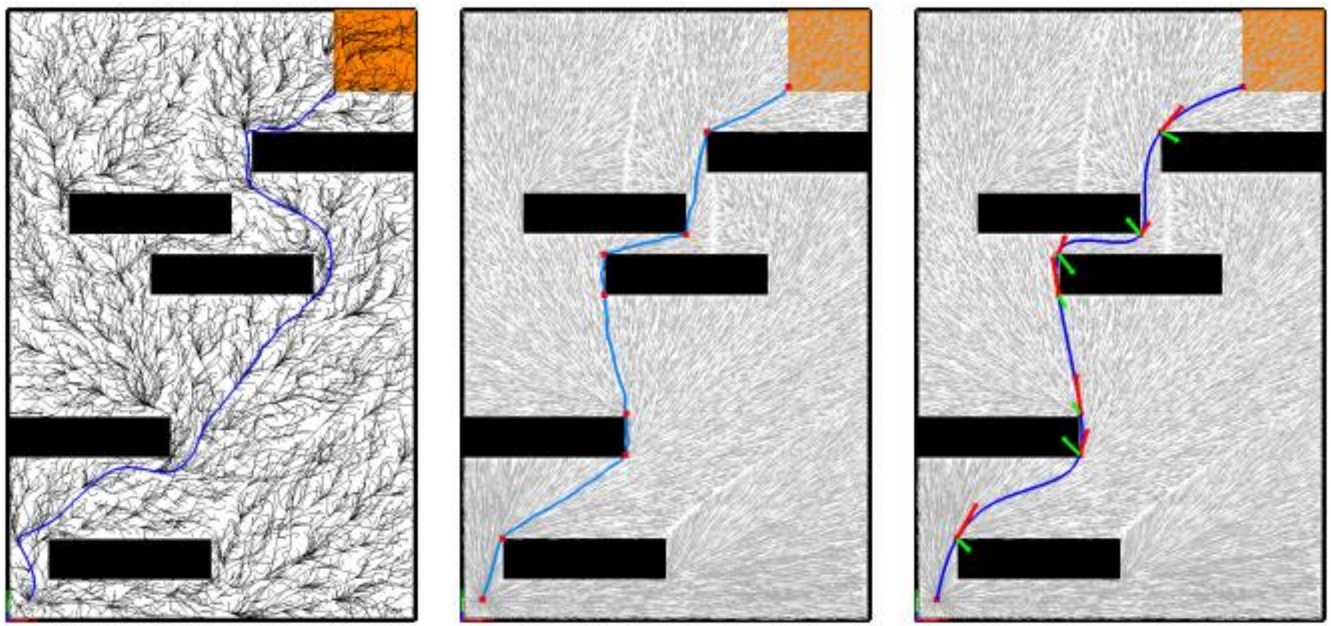
  Quadrotor : Minimize the snap! (second derivative of acceleration)

  (Just based on D.Mellinger & V.Kumar's paper. See if you wonder)
- Trajectory refinement (with time)
- Trajectory refinement (with max velocity)
- Trajectory refinement (with state constraints)
- Trajectory refinement (handling collision)

D. Mellinger and V. Kumar, " Minimum Snap Trajectory Generation and Control for Quadrotors ," in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), May 2011

Pathplanning

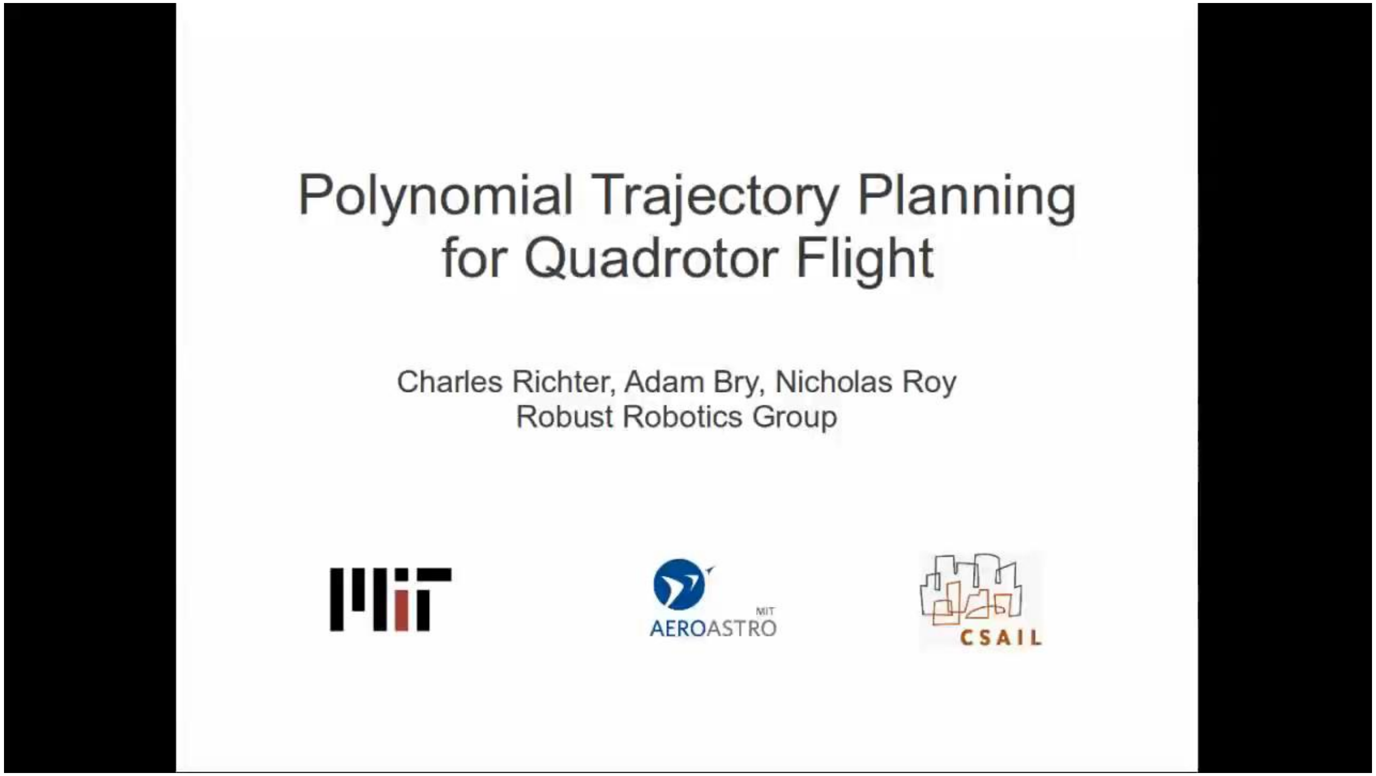
- Based on RRT* (Reference paper)



C. Richter, A. Bry, and N. Roy, “Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments,” in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2013.

Pathplanning

- Result of reference paper



Method	Runtime	$J_{poly.}$	T_{path}	L_{path}
RRT* with Polynomial Steer Function	120s	5.72×10^8	21.94s	40.35m
Low-Dim. Search + Unconstrained QP Optimization	3s	1.07×10^5	19.66s	35.51m

C. Richter, A. Bry, and N. Roy, “Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments,” in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2013.

Pathplanning

- Consider Cost!

  Sacrifice theoretical optimality but get good computation time

  - Quadrotor : Minimize the snap! (second derivative of acceleration)
  - Trajectory Polynomial

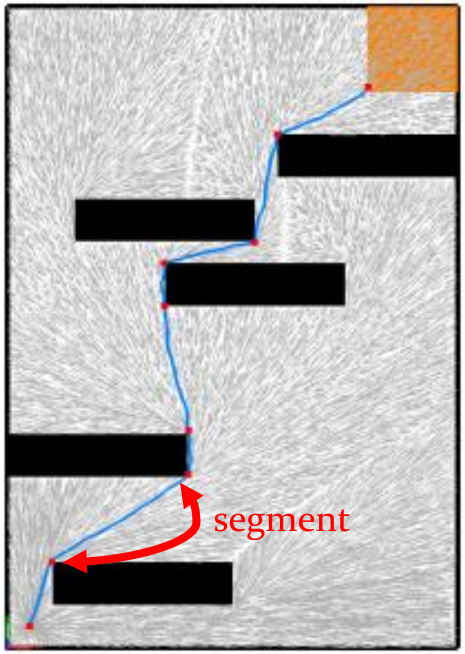
$$p(t) = \mathbf{t} \cdot \mathbf{c};\ \mathbf{t} = \begin{bmatrix} 1 & t & t^2 \ldots t^{N-1} \end{bmatrix};\mathbf{c} = \begin{bmatrix} c_0 \ldots c_{N-1} \end{bmatrix}^T$$

  - Cost over Trajectory

$$J_{\text{polynomials}} = \sum_{i=1}^{M} \sum_{d=1}^{D} \overbrace{\int_0^{T_{s,i}} \underbrace{\sum_{j=0}^{N-1} \left\| \frac{d^j p_{i,d}(t)}{dt^j} \right\| \cdot w_j}_{\text{cost per derivative term}}}^{J_{i,d}\text{,cost per polynomial}}$$

M segment, D dimension (So, each segment have D polynomials)

The only $\omega_4 = 1$ ($other\ \omega = 0$) Due to focus on just 'snap'



Pathplanning

- Consider Cost!
  - Segment's cost

$$J_{i,d} = \boldsymbol{c}_{i,d}^T \cdot \boldsymbol{Q}(T_{s,i}) \cdot \boldsymbol{c}_{i,d}$$

  - Segment's coefficient

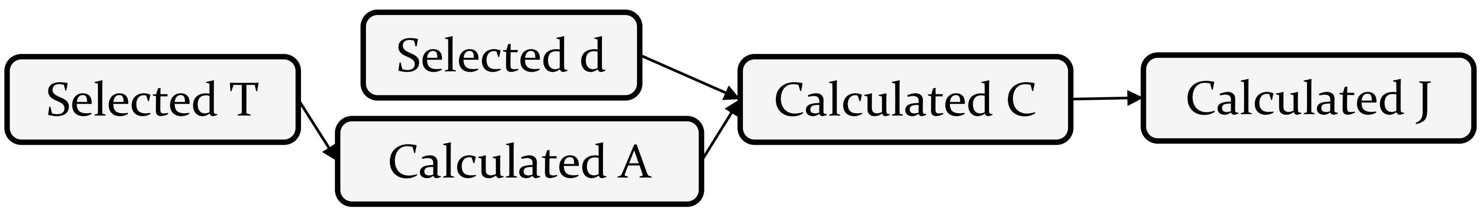
$$\underbrace{\begin{bmatrix} \boldsymbol{d}_{i,d,\text{start}} \\ \boldsymbol{d}_{i,d,\text{end}} \end{bmatrix}}_{\boldsymbol{d}_{i,d}} = \underbrace{\begin{bmatrix} \boldsymbol{A}(t=0) \\ \boldsymbol{A}(t=T_{s,i}) \end{bmatrix}}_{\boldsymbol{A}} \cdot \boldsymbol{c}_{i,d}$$

To get c, we have derivative value of polynomial.
We must have N coefficient (c is N×N matrix), so we only need N/2 amount of derivative for each start and end points.

$$\boldsymbol{A}(t=0) = \left[\frac{d^0}{dt^0}\boldsymbol{t}(0)^T \dots \frac{d^{N/2-1}}{dt^{N/2-1}}\boldsymbol{t}(0)^T \right]^T$$

$$\boldsymbol{A}(t=T_{s,i}) = \left[\frac{d^0}{dt^0}\boldsymbol{t}(T_{s,i})^T \dots \frac{d^{N/2-1}}{dt^{N/2-1}}\boldsymbol{t}(T_{s,i})^T \right]^T$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}(t=0) \\ \boldsymbol{A}(t=T_{s,i}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{0} \\ \boldsymbol{\Gamma} & \boldsymbol{\Delta} \end{bmatrix}$$



Pathplanning

- Consider Cost!
  - How to get free/unspecified derivatives?(In reference paper)

$$J = \begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_M \end{bmatrix}^T \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_M \end{bmatrix}^{-T} \begin{bmatrix} Q_1 & & \\ & \ddots & \\ & & Q_M \end{bmatrix} \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_M \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_M \end{bmatrix}$$

$$J = \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix}^T \underbrace{CA^{-T}QA^{-1}C^T}_{R} \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix} = \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix}^T \begin{bmatrix} R_{FF} & R_{FP} \\ R_{PF} & R_{PP} \end{bmatrix} \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix}$$

d_F : *Fixed derivatives*
d_P : *Free derivatives*
C : *permutation matrix assembled of* 0,1

$$J = \mathbf{d}_F^T R_{FF} \mathbf{d}_F + \mathbf{d}_F^T R_{FP} \mathbf{d}_P + \mathbf{d}_P^T R_{PF} \mathbf{d}_F + \mathbf{d}_P^T R_{PP} \mathbf{d}_P$$

Differentiating J
Equating to zero

$$\mathbf{d}_P^* = -R_{PP}^{-1} R_{FP}^T \mathbf{d}_F$$

C. Richter, A. Bry, and N. Roy, "Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2013.

Pathplanning

- Trajectory refinement (with time)

$$J = J_{\text{polynomials}} + k_T \cdot (\sum_{i=1}^{M} T_{s,i})^2$$

- Trajectory refinement (with max velocity)

$$v_{\text{norm}}(t) = \sqrt{(\dot{p}(t)_x)^2 + (\dot{p}(t)_y)^2 + (\dot{p}(t)_z)^2}$$

$$\frac{dv_{\text{norm}}(t)}{dt} = \frac{2\,(\dot{p}(t)_x \cdot \ddot{p}(t)_x + \dot{p}(t)_y \cdot \ddot{p}(t)_y + \dot{p}(t)_z \cdot \ddot{p}(t)_z)}{-\sqrt{(\dot{p}(t)_x)^2 + (\dot{p}(t)_y)^2 + (\dot{p}(t)_z)^2}}$$

$$\boldsymbol{t} \cdot (\dot{\boldsymbol{c}}_x * \ddot{\boldsymbol{c}}_x) + \boldsymbol{t} \cdot (\dot{\boldsymbol{c}}_y * \ddot{\boldsymbol{c}}_y) + \boldsymbol{t} \cdot (\dot{\boldsymbol{c}}_z * \ddot{\boldsymbol{c}}_z) \stackrel{!}{=} 0$$

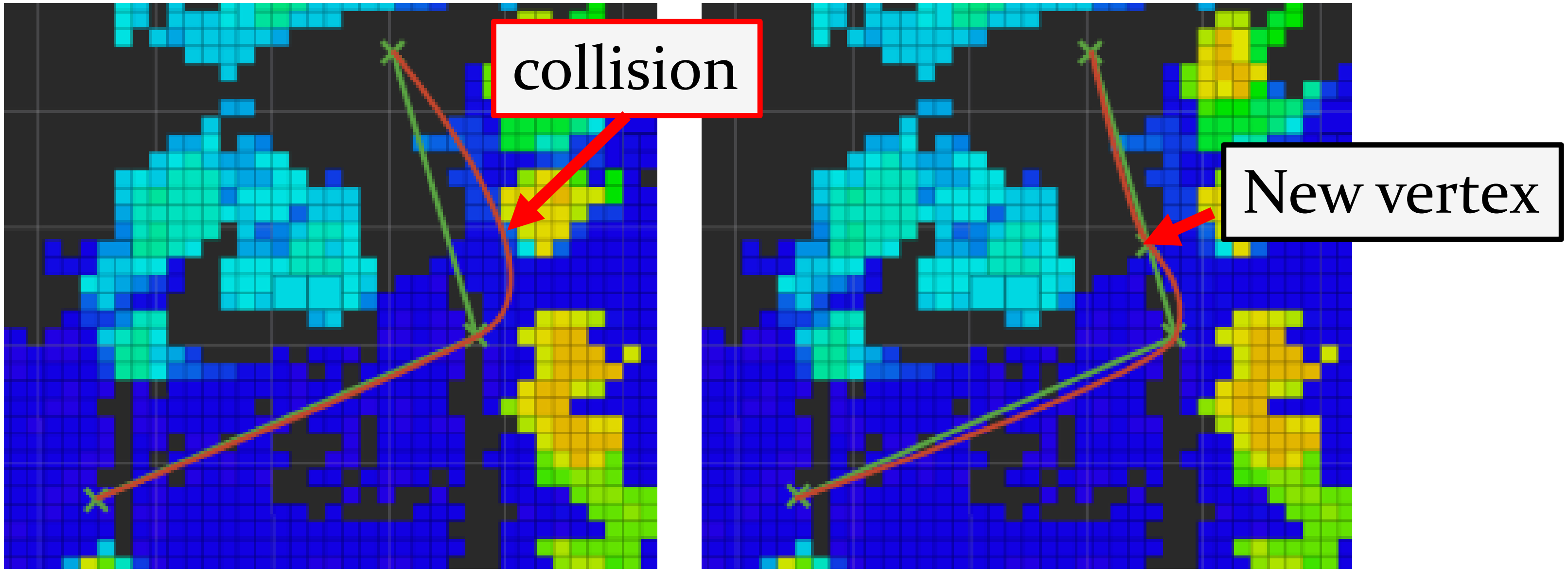
$$\boldsymbol{t} \cdot (\dot{\boldsymbol{c}}_x * \ddot{\boldsymbol{c}}_x + \dot{\boldsymbol{c}}_y * \ddot{\boldsymbol{c}}_y + \dot{\boldsymbol{c}}_z * \ddot{\boldsymbol{c}}_z) \stackrel{!}{=} 0$$

Pathplanning

- Trajectory refinement (with state constraints)

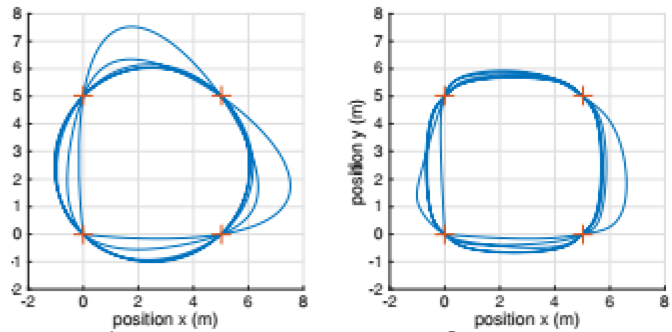
$$J_{\text{soft-constraint}} = \exp\left(\frac{x_{\text{max, actual}} - x_{\text{max}}}{x_{\text{max}} \cdot \epsilon} \cdot k_s\right)$$

- Trajectory refinement (handling collision)



Result

- Simple 4 points (5m x 5m)



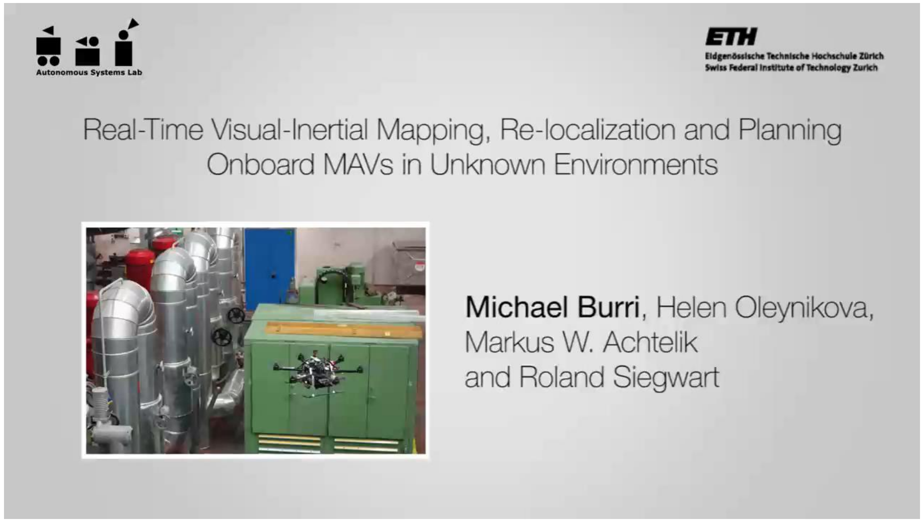
200s for initial trajectory -> 112s for optimized solution

- Time, success rate for amount of segment

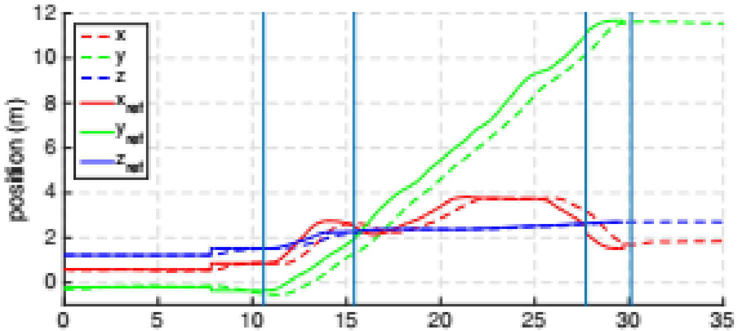
segments	t_{init} (ms)	t_{opt} (ms)	std dev t_{opt} (ms)	success (%)
3	0.117	48.0	12.1	96
5	0.171	143	41.4	95
10	0.297	584	169	91
20	0.565	2157	632	88
50	1.58	10110	1290	47

success : does not exceed the state limits by 10%

Result



- Time, success rate for amount of segment



Bold : planned, dot : followed

Discussion

- In large environment, the method in this paper may take less time than sampling based method.
- There aren't any compare in this paper. So we don't know any performance of the method.
- Can choose the pose at each point, so It can be applied in various situation (in this paper, they set to see always in direction of flying to avoid active obstacle.