SplatNav :
Safe Real-Time Robot Navigation in Gaussian Splatting Maps

2025. 11. 26

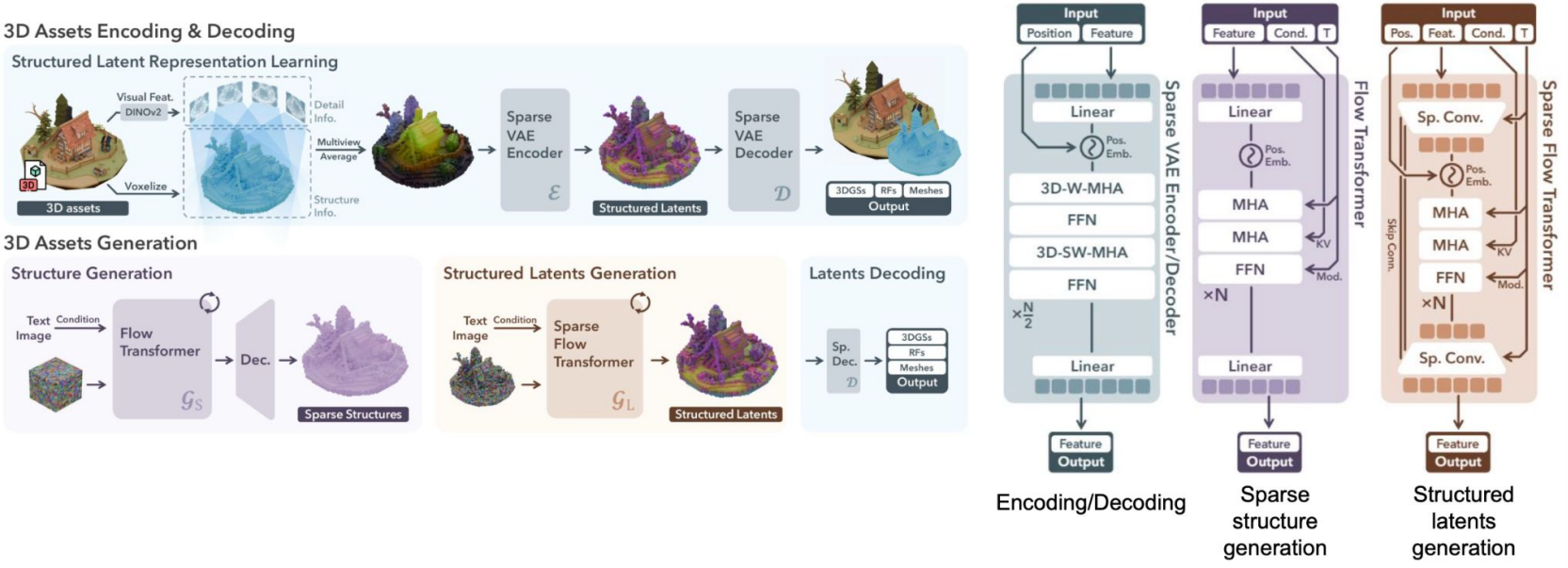
CS580 : Computer Graphics
Team #3 Paper Presentation

Harin Kim & Jiwon Park

KAIST

Review of Previous Lecture

The Network Structures for Encoding, Decoding, and Generation



Motivation

Why do we use Gaussian Splatting for Navigation task?

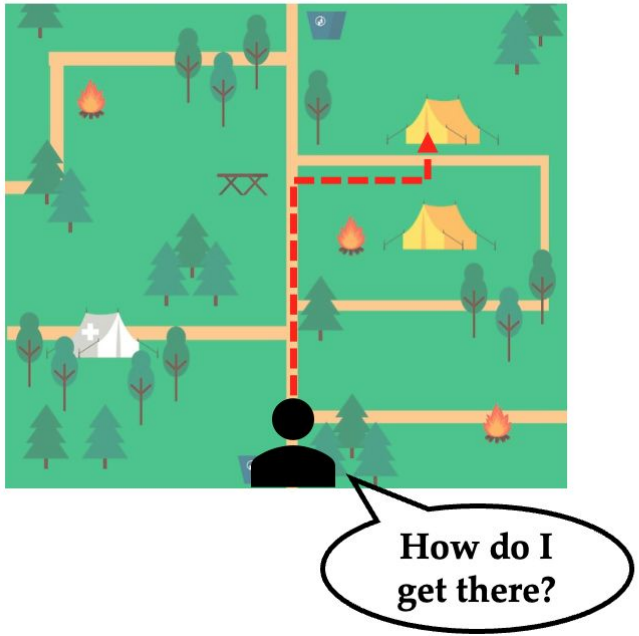
- Traditional map representation (occupancy grid, mesh, SDF, point cloud) has resolution restrictions & loss of geometry information
- NeRF-based navigation is slow & non real-time

Gaussian Splatting

= High quality geometry + Fast rendering + collision detection utilizing ellipsoids

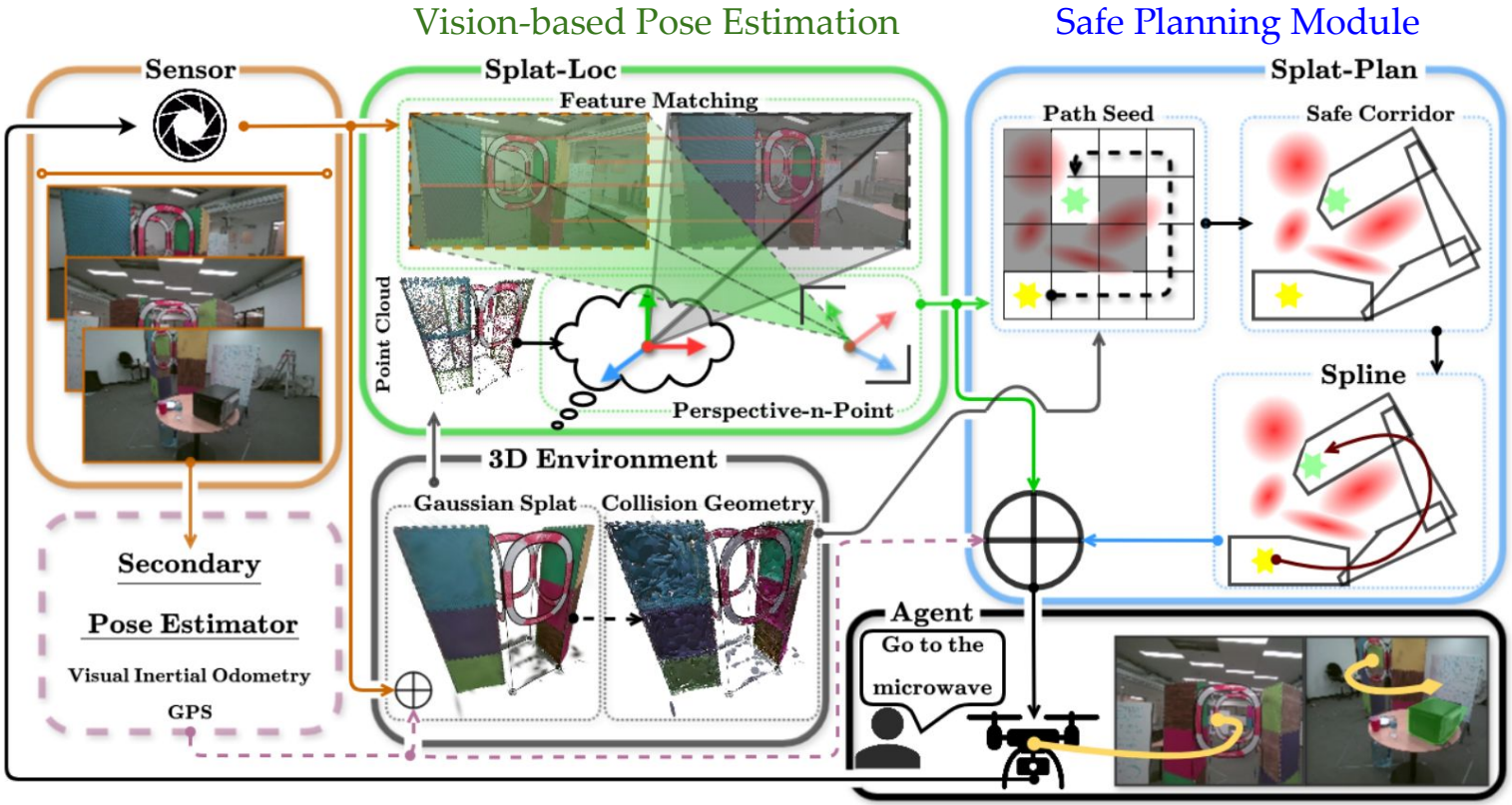
Map-based Navigation Pipeline

Path planning



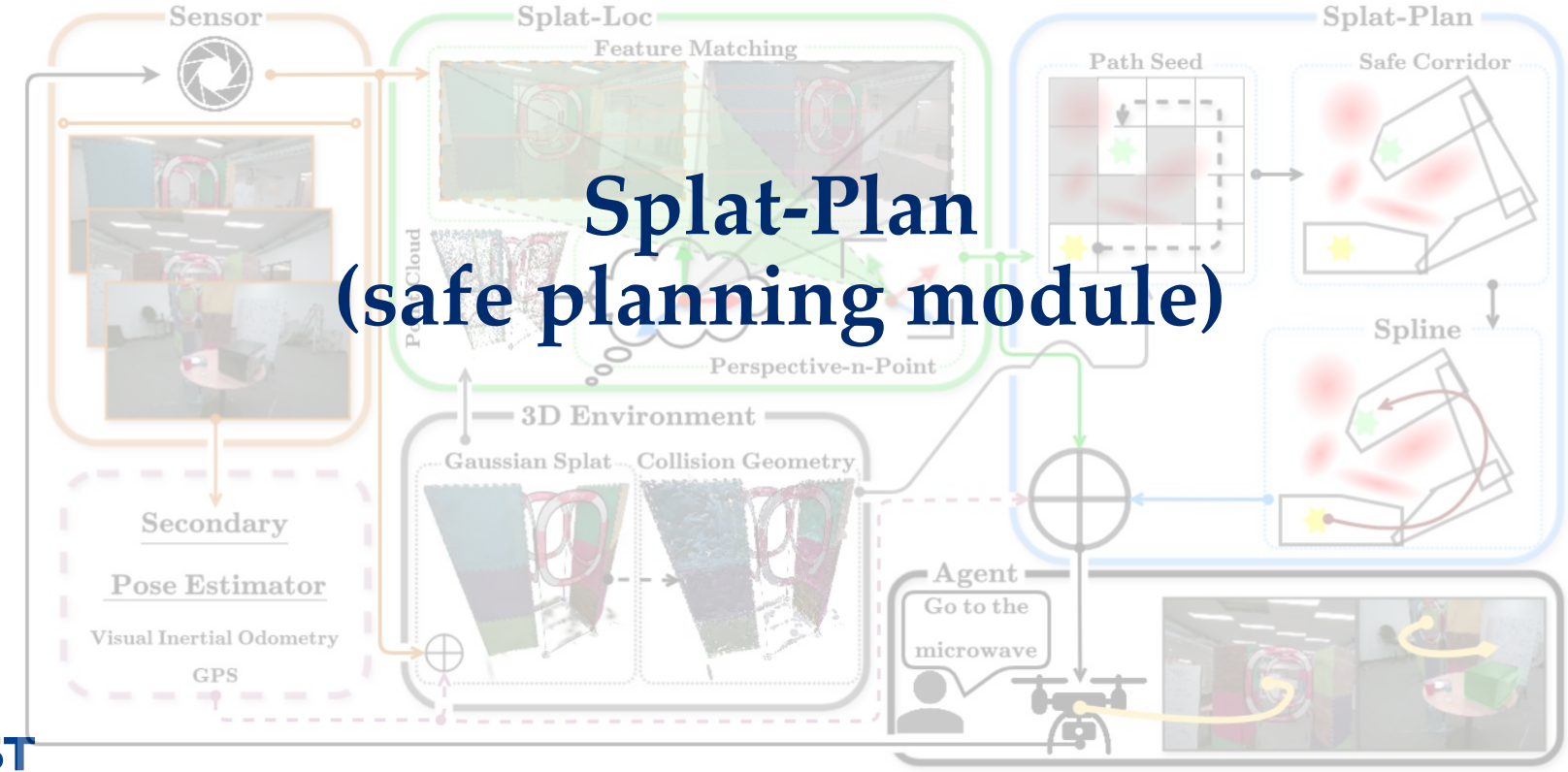
Localization

Splat-Nav : System Overview



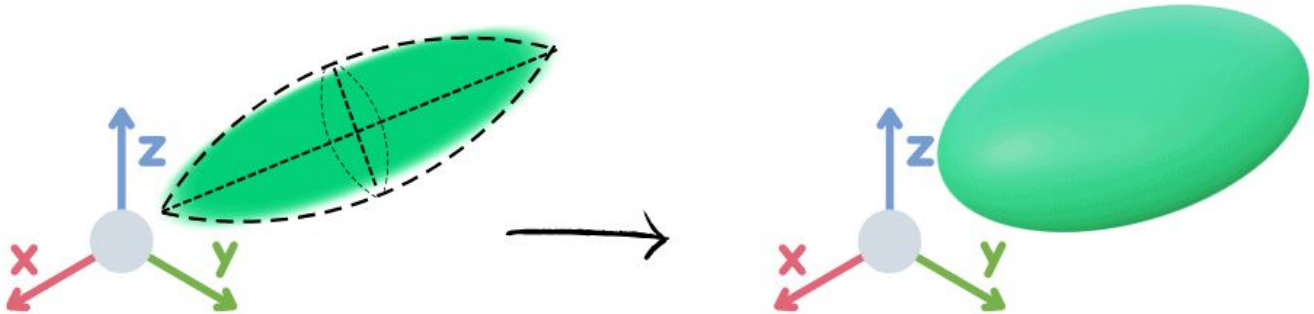
KAIST

Splat-Plan
(safe planning module)



KAIST

3D gaussian → 3D ellipsoids



3D Gaussian *3D Ellipsoid*

Occupancy - Ellipsoidal Representation

- Robot ($\mathcal{R}$) : Modeled as an ellipsoid
- Map ($\mathcal{G}$): Set of all Gaussians in the GSplat

$$\mathcal{G} = \{\varepsilon_j\}_{j=1}^{N}$$

Occupied space in environment with γ% confidence :

$$\varepsilon_j = \{x \in \mathbb{R}^3 \,|\, (x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j) \leq \chi_3^2(\gamma)\}$$

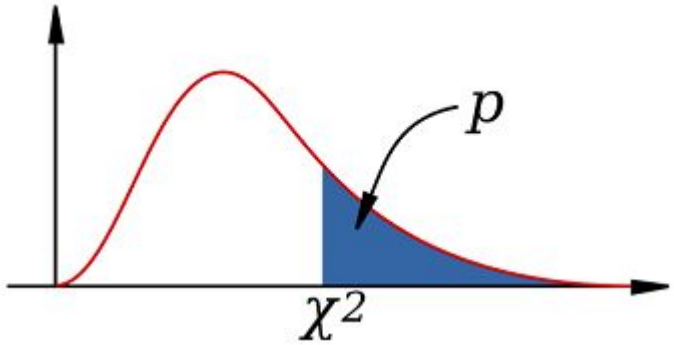
Mean of gaussian

γth percentile of chi-square distribution

KAIST

Occupancy - Ellipsoidal Representation

$\mathcal{G} = \{\mathcal{E}_j\}_{j=1}^{N}$



Occupied space in environment with γ% confidence :

$$\mathcal{E}_j = \{x \in \mathbb{R}^3 \,|\, (x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j) \leq \chi_3^2(\gamma)\}$$

Mean of gaussian

γth percentile of chi-square distribution

KAIST

Occupancy - Ellipsoidal Representation

$$\mathcal{G} = \{\mathcal{E}_j\}_{j=1}^{N}$$

Occupied space

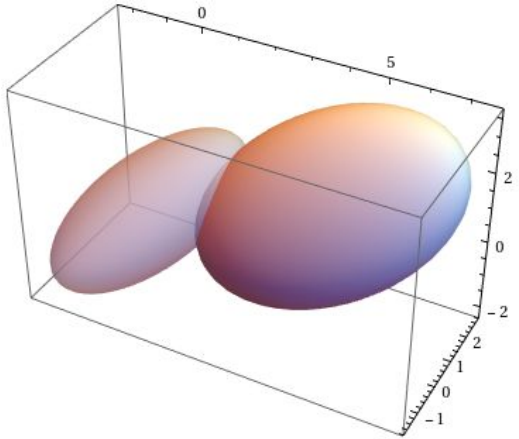
$$\mathcal{E}_j = \{x \in \mathbb{R}^3 \,|\, (x-\mu_j)^T \Sigma_j^{-1} (x-\mu_j) \leq 1\}$$

Math : Ellipsoid Intersection Test

Theorem 1 : Two ellipsoids $\mathcal{E}_a$ and $\mathcal{E}_b$ are disjoint if and only if there exists a scalar $s \in (0,1)$ such that :

$$K(s) > 1 \qquad \text{where}$$

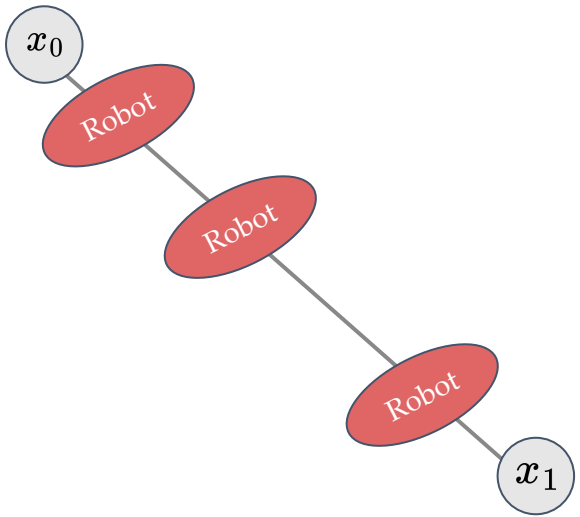
$$K(s) = (\mu_b - \mu_a)^T \Sigma_{a,b}^{-1}(s)(\mu_b - \mu_a)$$



→ Concave in s and convex with respect to the means & variances!

Extension to Linear Motion

Starting point



Ending point

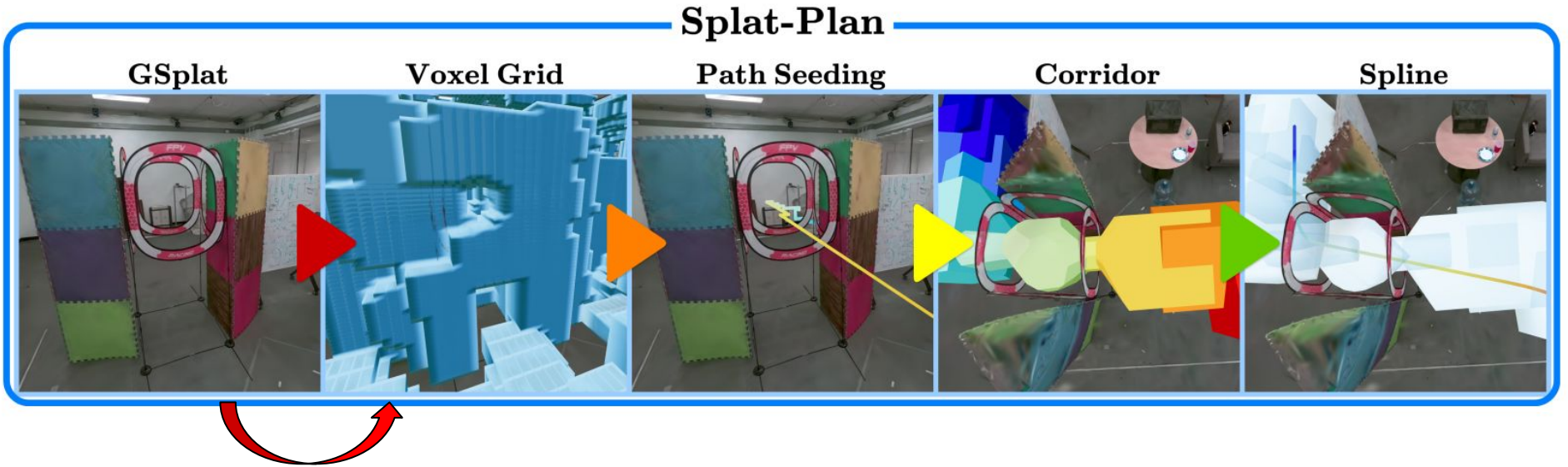
Linear segment : $l(t) = x_0 + t(x_1 - x_0)$

Final Safety Test

$$\min_{t\in[0,1]} \max_{s\in(0,1)} K(s,t) > 1$$

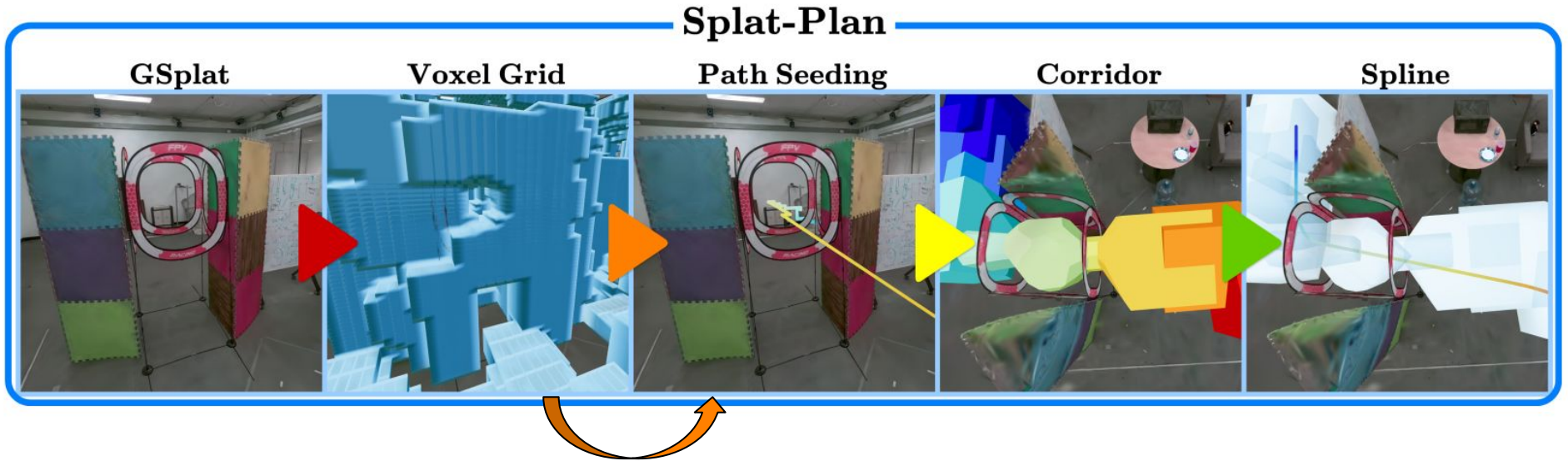
Splat-Plan

- A safe-planning module



Splat-Plan

- A safe-planning module

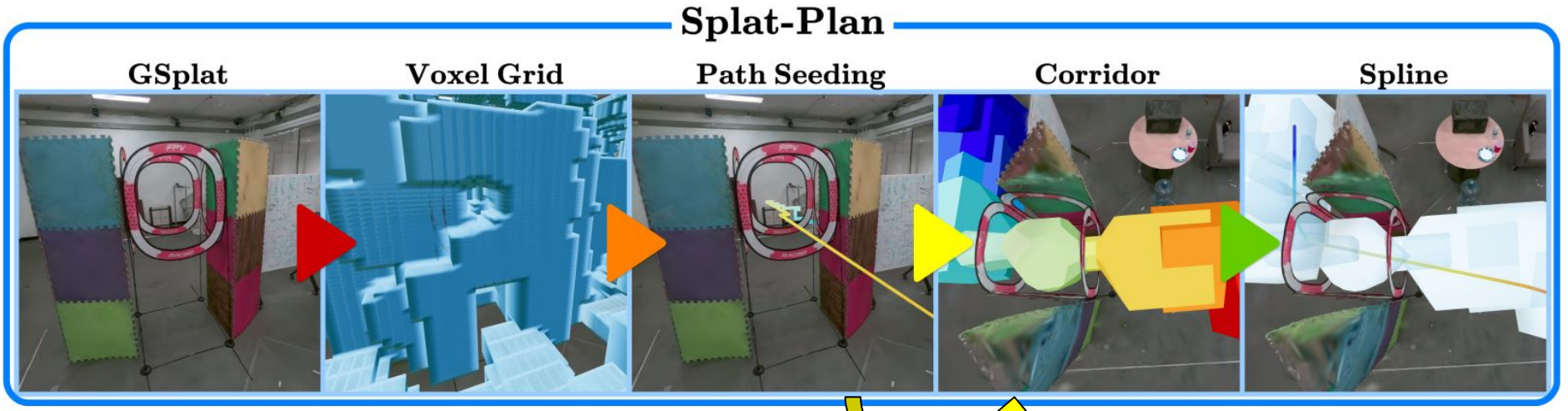


Creates safe but non-smooth path

Dijkstra algorithm

Splat-Plan

- A safe-planning module

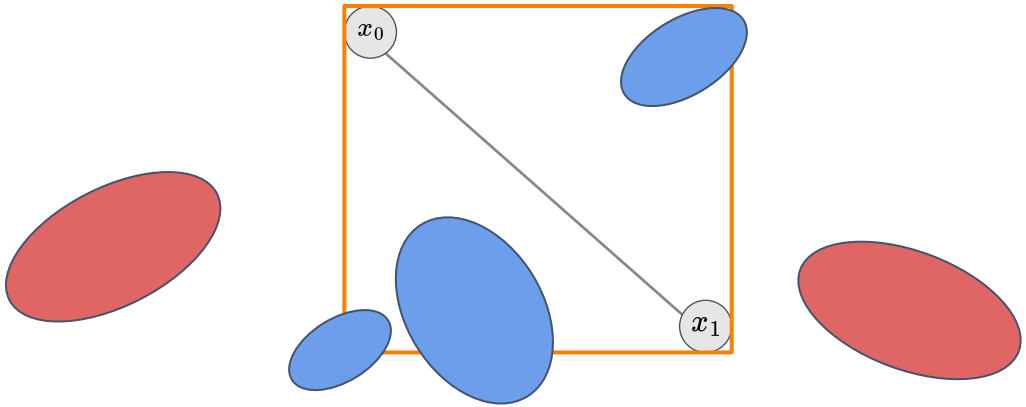


1. Find collision set
2. Polytope generation

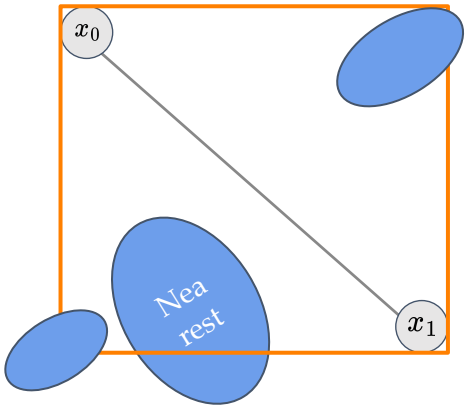
Form a corridor around the feasible path

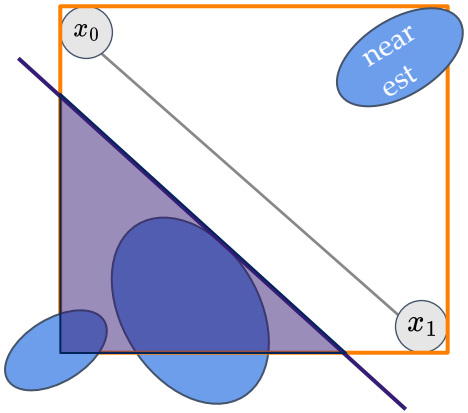
Find Collision Set

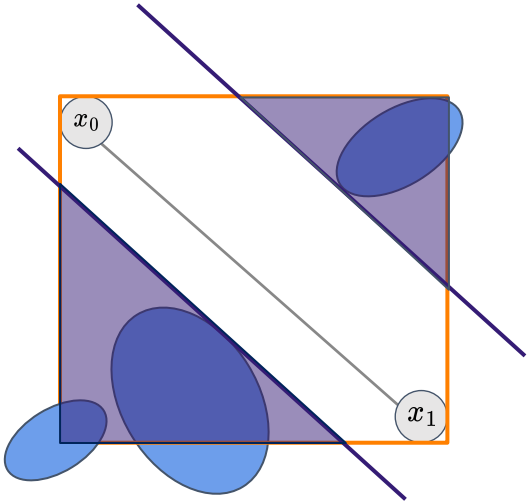
- Iterating through the line segments in the seed path…



Polytope Generation

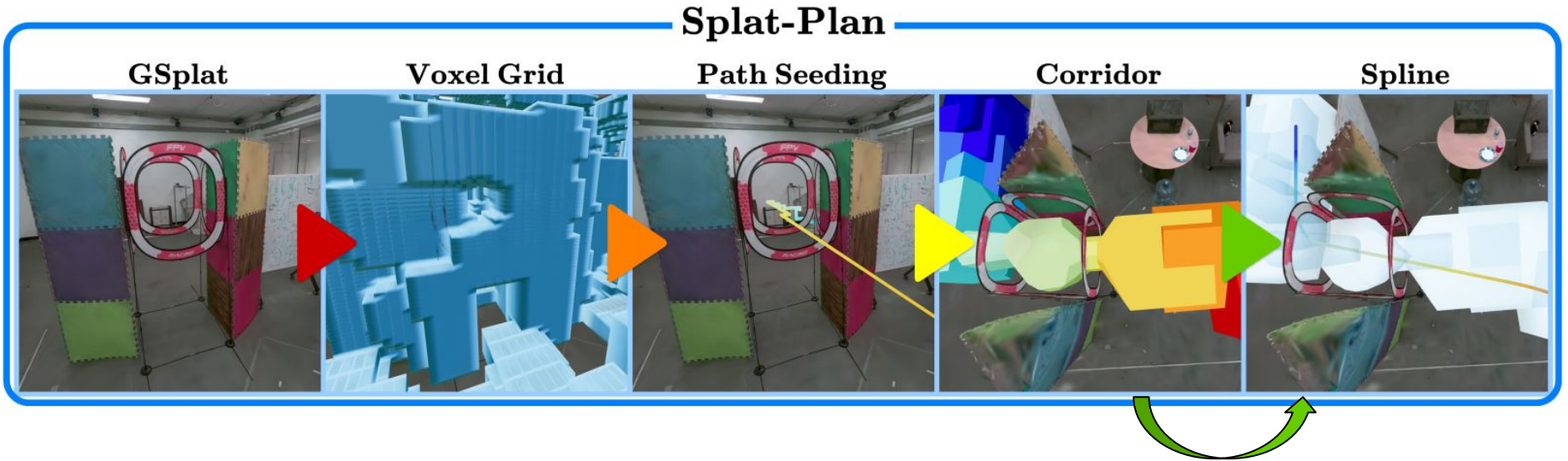






Splat-Plan

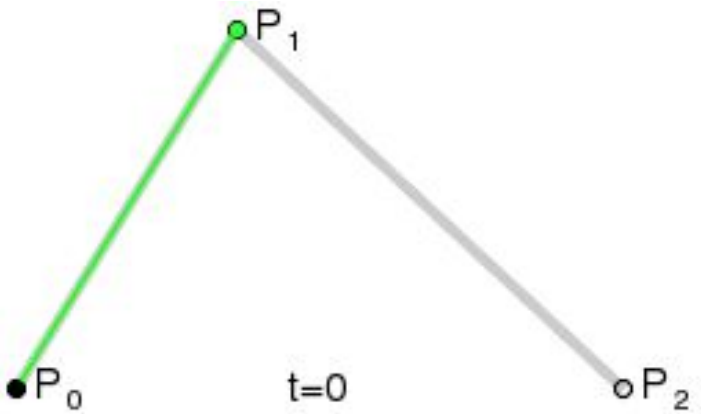
- A safe-planning module



Solve a quadratic program to achieve optimized Bezier splines

Trajectory Optimization : Bézier curve

- 3 control points : P_0, P_1, P_2



[Convex Hull Property]

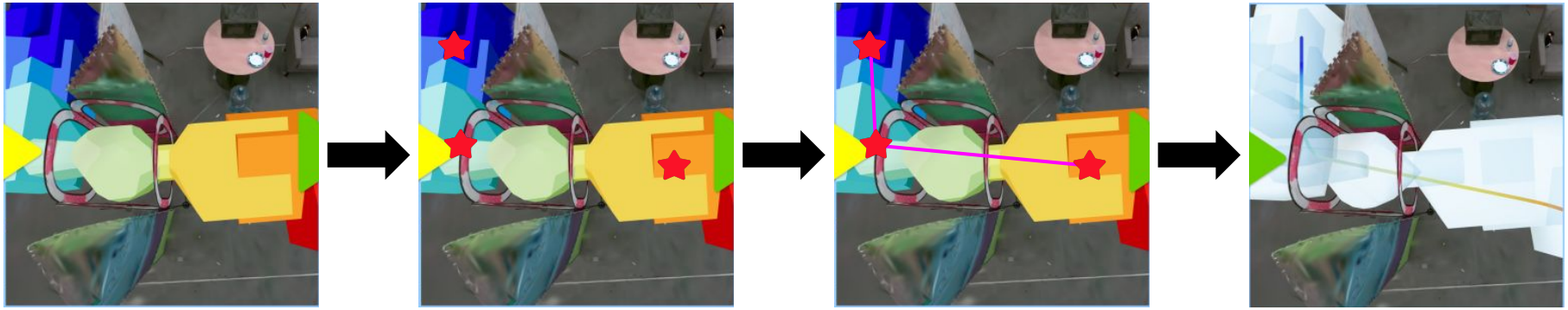
If all the control points of a Bezier curve are inside a convex shape, then the entire curve is guaranteed to be inside that shape.

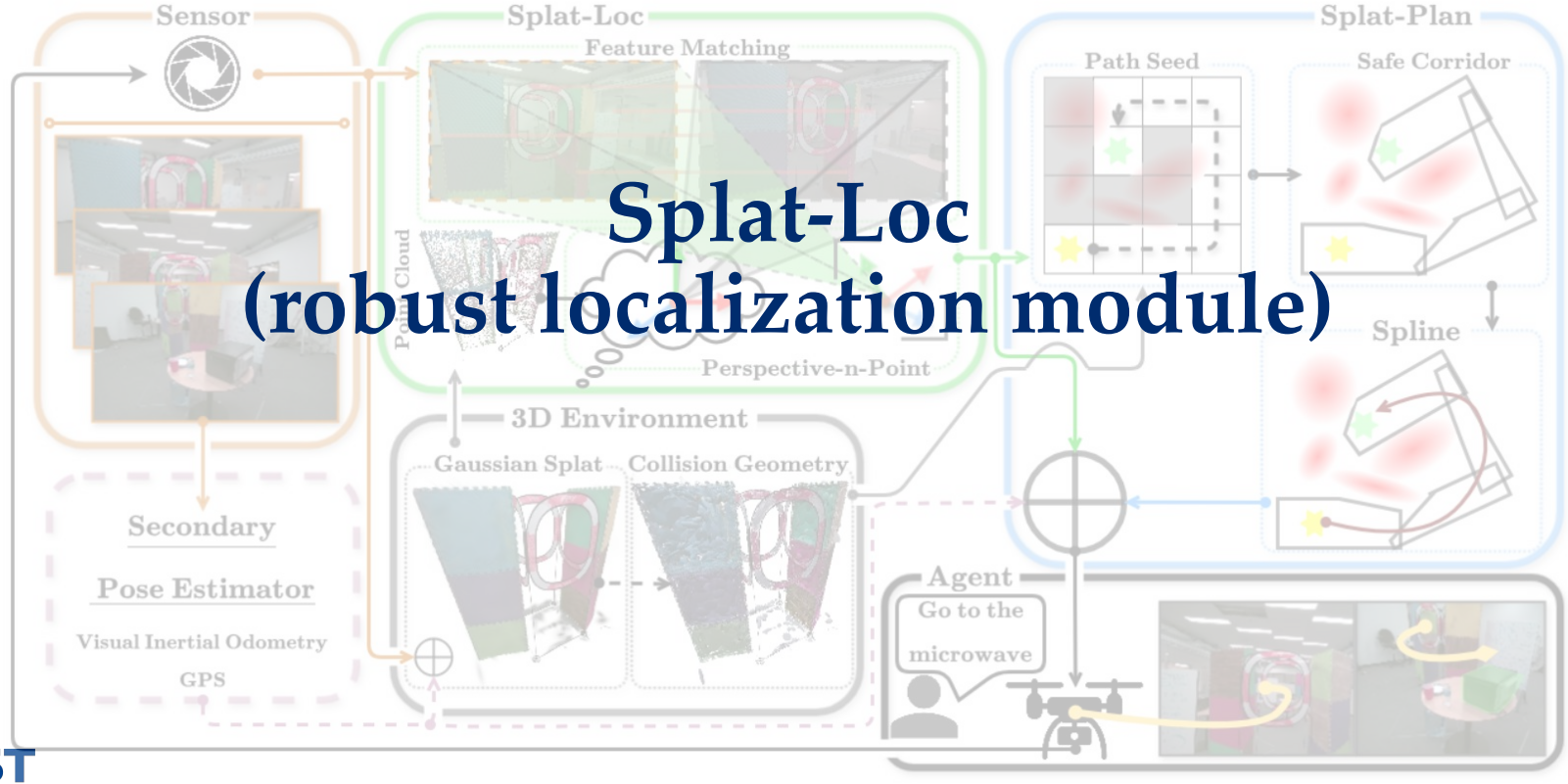
Trajectory Optimization : Bézier curve

- Goal : Find the best locations for these control points!
  - Minimize squared distance between consecutive contro points

- Formulated as **Quadratic Program (QP)**
  - Can be solved in real-time

Trajectory Optimization





Splat-Loc
(robust localization module)

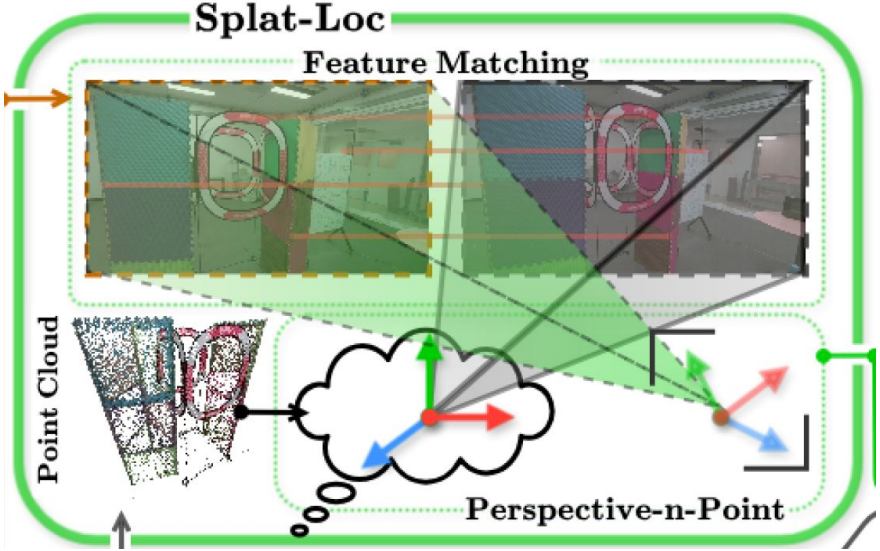
KAIST

Splat-Loc

- Lightweight **Monocular Pose Estimation**
  - Performs real-time recursive state estimation using only RGB camera
  - Leverages GSplat scene representation for localization
  - Provides high-frequency pose updates (~25Hz) to the planning module

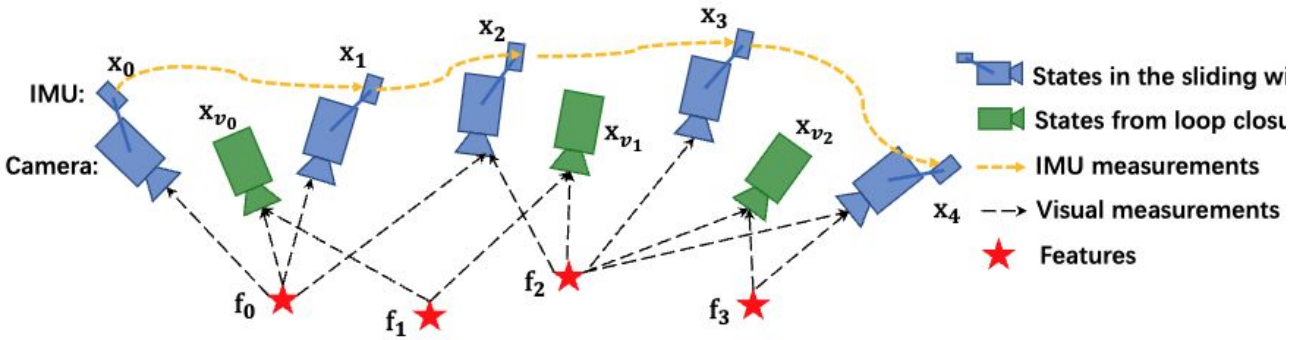
"Which **pixel in the real image** corresponds to which **3D point in the GSplat world**?"
(Feature matching for 2D-3D correspondences)

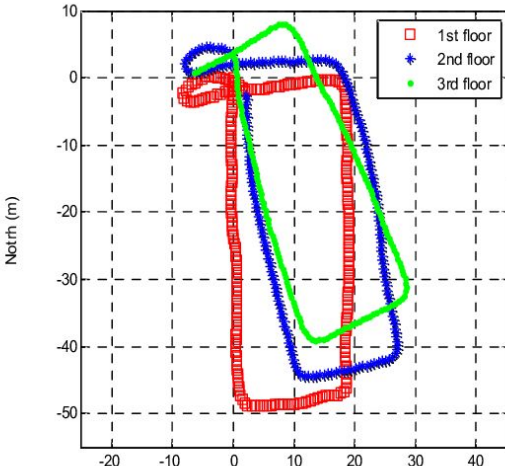
"Find the camera pose T that makes the projected 3D points land as close as possible to the real-image pixels"
(Reprojection + PnP optimization)



Splat-Loc

- Traditional Monocular Image based Pose Estimation
  - VIO (Visual-Inertial Odometry)
    - Fuses camera + IMU
    - Provides metric scale, handles fast motion and high-frequency dynamics

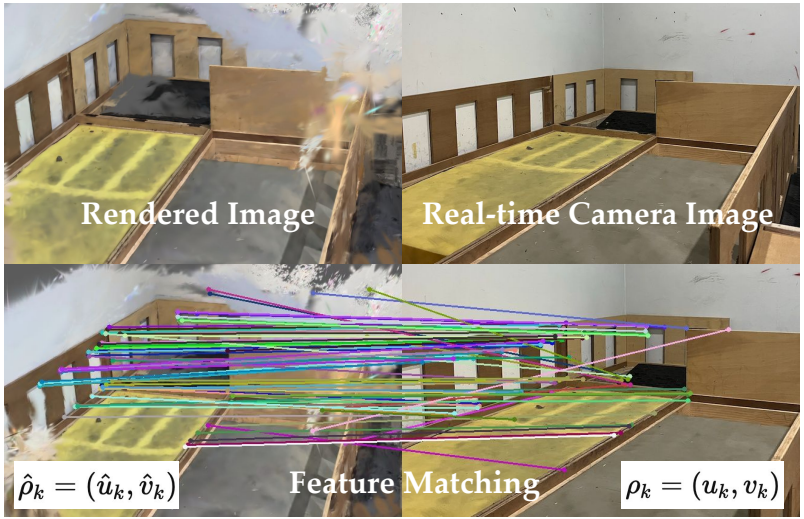




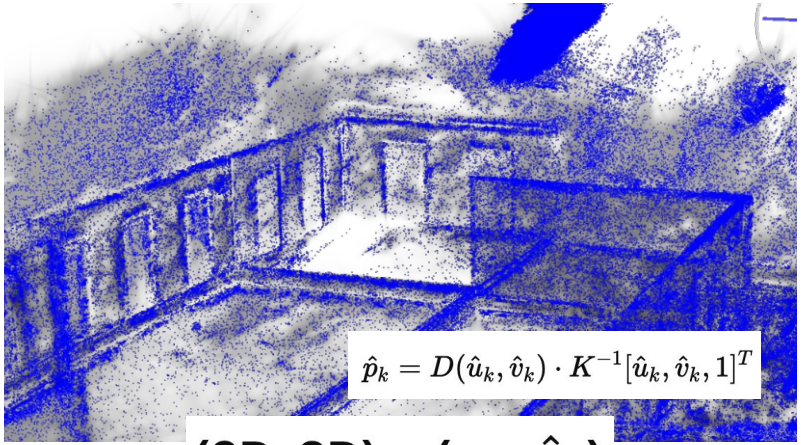
VINS-mono (VIO using monocular camera)

Splat-Loc

- Pipeline
  - **Render-and-Match** strategy
    - Rendering : Renders an RGB image using **pose estimation**
    - Feature Extraction : utilizes LightGlue to extract keypoints and descriptors from **both camera image and the rendered image**



2D-2D pairs + Camera Intrinsic

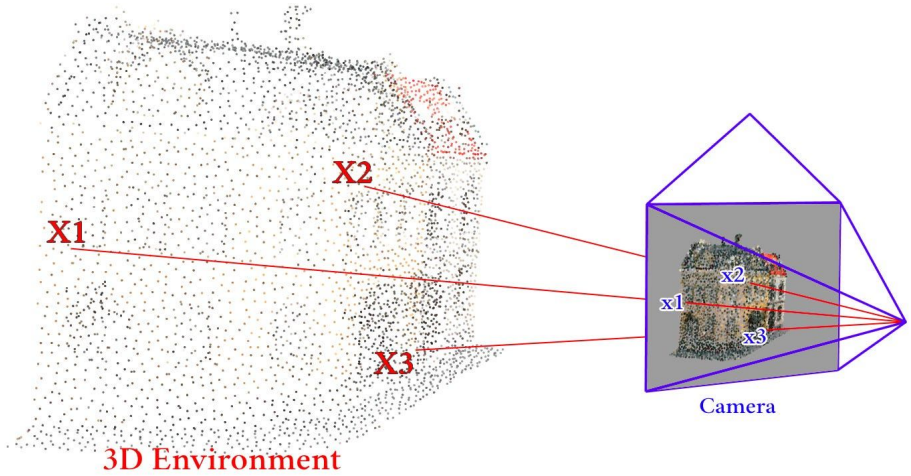


$$(2D, 3D) = (\rho_k, \hat{p}_k)$$

Pixel coordinate 3D coordinate

Splat-Loc

- Perspective-n-Point (PnP)
  - Estimate a **camera's pose** (rotation + translation)
  - Uses **n 3D world points** and their corresponding **2D image projections**
  - Solves for the pose that minimizes reprojection error between observed 2D points and the projection of 3D points
  - Formulated as a non-linear least-square optimization problem



Splat-Loc

- Pipeline
  - Optimization via PnP
    - Formulates a Perspective-n-Point (PnP) problem using the matched 2D-3D correspondences
    - **Minimizes reprojection error** using the Levenberg-Marquardt algorithm with RANSAC for outlier rejection

Transformation (Pose)

$$\bar{T}_t = \text{argmin}_{T \in SE(3)} \sum_{k=1}^{\ell} \left|\left| \rho_k - K T \hat{p}_k \right|\right|^2$$

3D to 2D projection

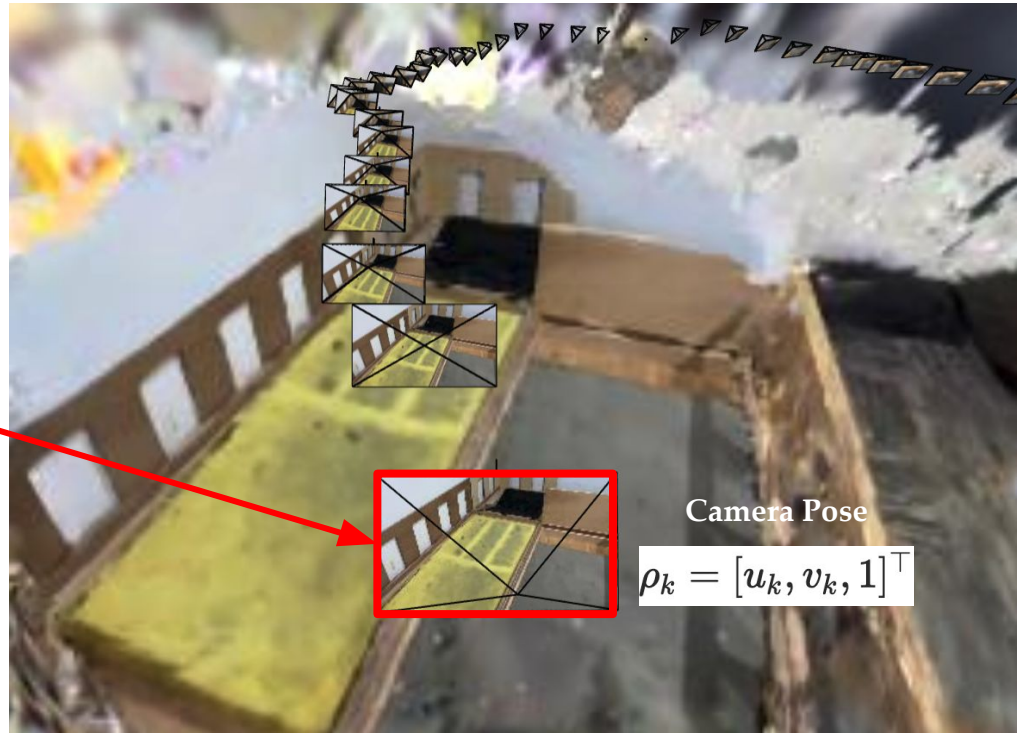
Pixel coordinate of matched feature in real image

3D world point from GSplat Map

$$\rho_k = [u_k, v_k, 1]^\top$$

Splat-Loc

- GS-Loc
  - Camera pose that minimizes **photometric error** between map rendering and actual camera image
- Splat-Loc
  - Minimizing **reprojection error**



Experiments

Splat-Plan

- Quantitative results
  - Comparison
    - NeRF-Nav, RRT*
  - SFC-1
    - Basic corridor constraint
    - Uses simple SDF-based corridor
  - SFC-2
    - Multi-segment corridor
    - Splits the global path into multiple linear segments
  - SFC-3
    - Corridor + Nonlinear optimization
    - Applies trajectory optimization inside the corridor
  - SFC-3
    - Corridor + Full optimization
    - Applies trajectory optimization inside the corridor

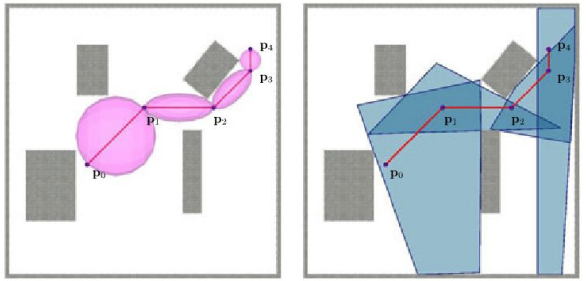


Fig. 5. Generate a *Safe flight corridor* (blue region) from a given path $P = \langle \mathbf{p}_0 \rightarrow \cdots \rightarrow \mathbf{p}_4 \rangle$. Left: find the collision-free ellipsoid for each line segment. Right: dilate each individual ellipsoid to find a convex polyhedron.

Safe Flight Corridor

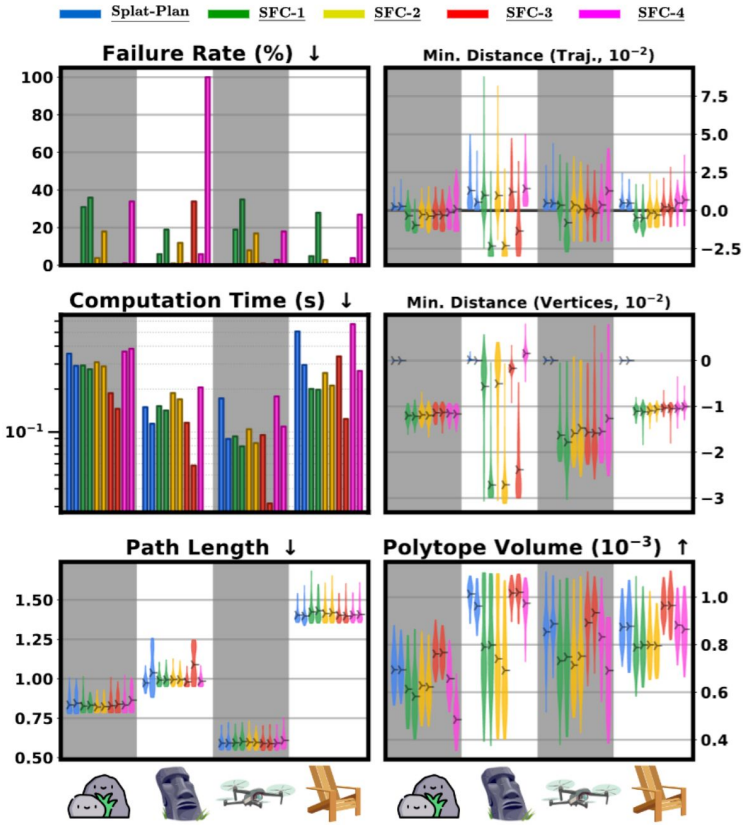
Splat-Plan

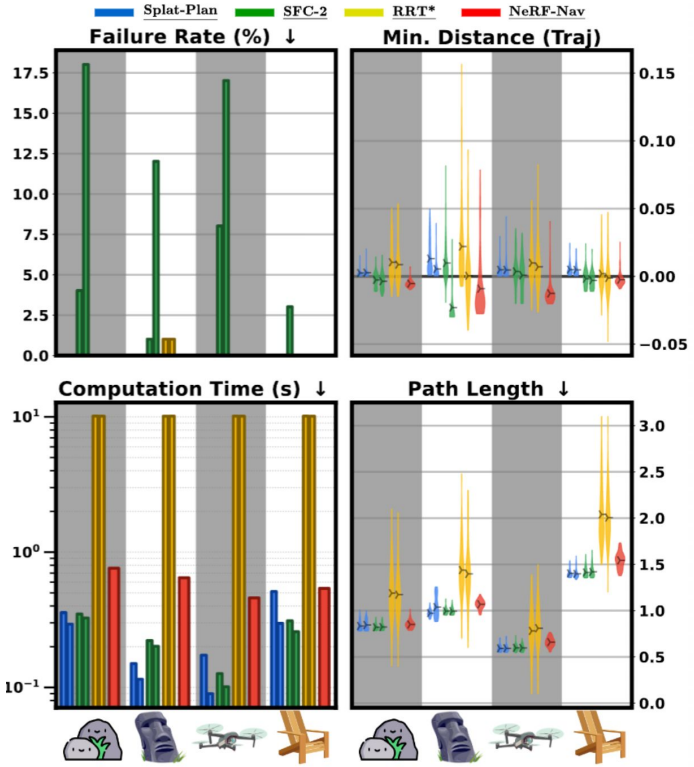
- Quantitative results
  - Comparison
    - NeRF-Nav, RRT*
  - Key metrics
    - Safety, Non-Conservativeness (NC), Smoothness, Feasibility, Realtime Execution(RT)
      - NC : measures whether the planner avoids being overly cautious
        - Check means the planner can safely pass near obstacles without taking long detours
      - Feasible : valid solution

	Safe	NC	Smooth	Feasible	RT	Env.
NeRF-Nav	×	N/A	×	✓	×	NeRF
RRT*	×	×	×	✓	×	GS
SFC-1	×	✓	✓	×	✓	GS
SFC-2	×	✓	✓	×	✓	GS
SFC-3	×	✓	✓	×	✓	GS
SFC-4	×	×	✓	×	✓	GS
Splat-Plan	✓	✓	✓	✓	✓	GS

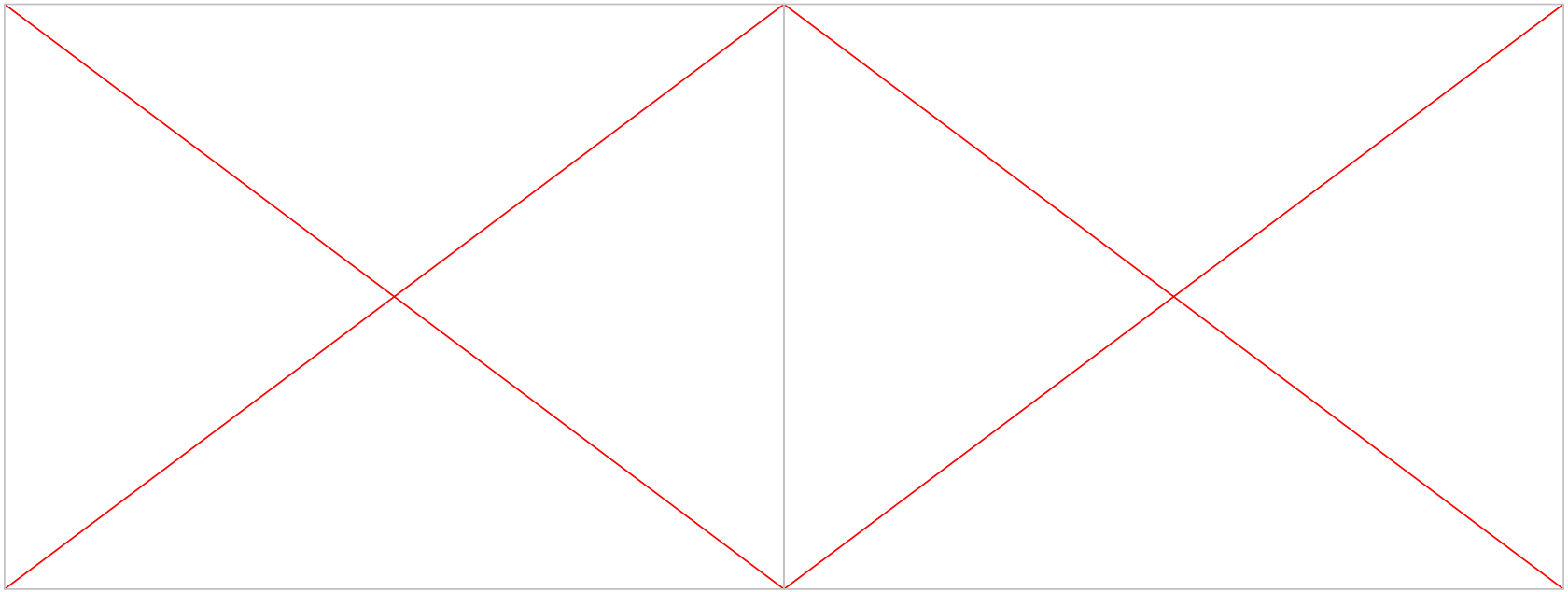
KAIST

Comparison


Splat-Plan
SFC-1
SFC-2
SFC-3
SFC-4
Failure Rate (%) ↓
Min. Distance (Traj., 10^{-2})
Computation Time (s) ↓
Min. Distance (Vertices, 10^{-2})
Path Length ↓
Polytope Volume (10^{-3}) ↑


Splat-Plan
SFC-2
RRT*
NeRF-Nav
Failure Rate (%) ↓
Min. Distance (Traj)
Computation Time (s) ↓
Path Length ↓

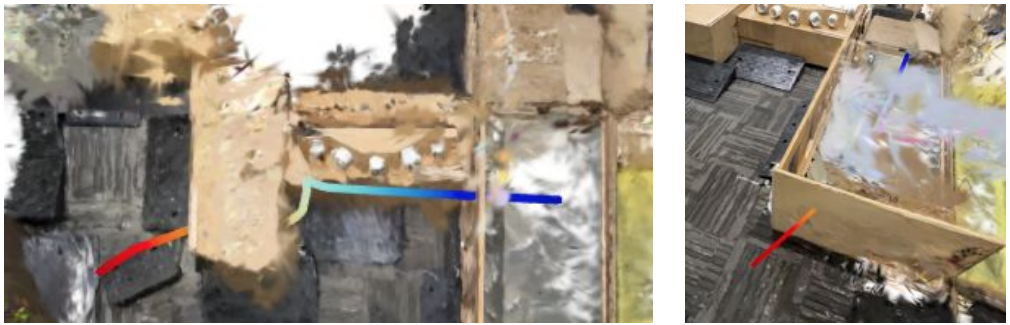
Qualitative results



KAIST

Splat-Plan

- Limitations
  - Dependence on **high-quality 3DGS maps**
    - Requires dense and well-reconstructed Gaussian fields
  - Limited robustness in **texture-poor or reflective areas**
    - 3DGS geometry may become inaccurate, affecting distance queries and collision checks
  - Assumes **static environments**
    - Dynamic obstacles are not modeled
  - Scalability challenges in large scenes
    - Querying large Gaussian sets can increase latency without spatial pruning



Failure cases in poor 3DGS maps

Splat-Loc

- Quantitative results
  - Tested on 4 environments
  - GT camera poses obtained from Blender (synthetic) or COLMAP (real scene)
  - Metric
    - Pose error measured as average pose difference in SE(3) manifold
  - Comparison
    - ICP
    - GS-Loc baseline (Using only photometric loss)

TABLE I: Comparison of baseline pose estimation algorithms in simulation in the **Statues** scene with $\delta_R = 20°$ and $\delta_t = 0.1\,\mathrm{m}$.

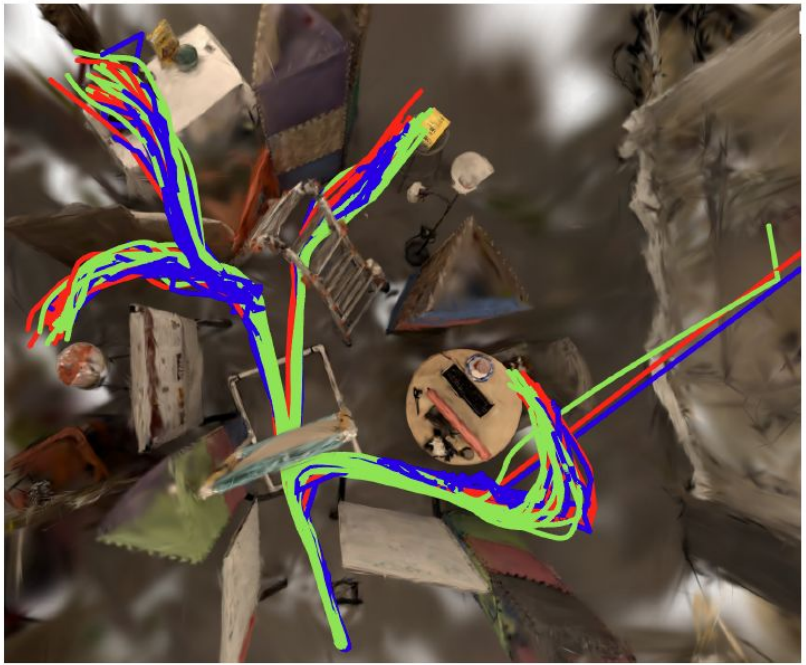
Algorithm	R.E. (deg.)	T.E. (cm)	C.T. (msec.)	S.R. (%)
ICP [59]	73.1 ± 45.9	129 ± 75	107 ± 19.2	100
Colored-ICP [58]	0.83 ± 0.37	1.31 ± 0.60	43.3 ± 9.70	100
Splat-Loc-SIFT (ours)	0.09 ± 0.06	0.56 ± 0.75	63.3 ± 2.39	100
Splat-Loc-Glue (ours)	$\mathbf{0.05 \pm 0.03}$	$\mathbf{0.33 \pm 0.27}$	$\mathbf{41.2 \pm 73.2}$	100
GS-Loc [34, 35, 36]	122 ± 33.8	245 ± 91.2	36200 ± 5440	100

TABLE II: Comparison of baseline pose estimation algorithms in the **Stonehenge** scene with $\delta_R = 30°$ and $\delta_t = 0.5\,\mathrm{m}$.

Algorithm	R.E. (deg.)	T.E. (cm)	C.T. (msec.)	S.R. (%)
ICP [59]	131 ± 22.6	370 ± 554	122 ± 153	100
Colored-ICP [58]	94.9 ± 51.3	57.4 ± 28.8	488 ± 104	20
Splat-Loc-SIFT (ours)	$\mathbf{0.217 \pm 0.0369}$	0.334 ± 0.0563	139 ± 3.15	70
Splat-Loc-Glue (ours)	0.220 ± 0.203	$\mathbf{0.315 \pm 0.210}$	$\mathbf{45.1 \pm 0.611}$	100

Splat-Loc

- Qualitative results



Motion capture (GT)

Splat-Loc

VIO

Splat-Loc

- Advantages & Disadvantages

Simple pipeline	No descriptor training, no 2D-3D feature matching
Integration with 3DGS renderer	The same pipeline used for Splat-plan

Advantages

Strong initial-pose dependence	Optimization is local → Diverges if initial guess is far from the true pose
Finite-difference gradient is expensive	Requires many renderings per iteration → Slow for large Gaussian maps
Prone to local minima	In texture-less scenes, repetitive patterns, Low gradient regions
Not for global relocalization	No retrieval mechanism

Disadvantages

Q&A

KAIST

Quiz :

KAIST