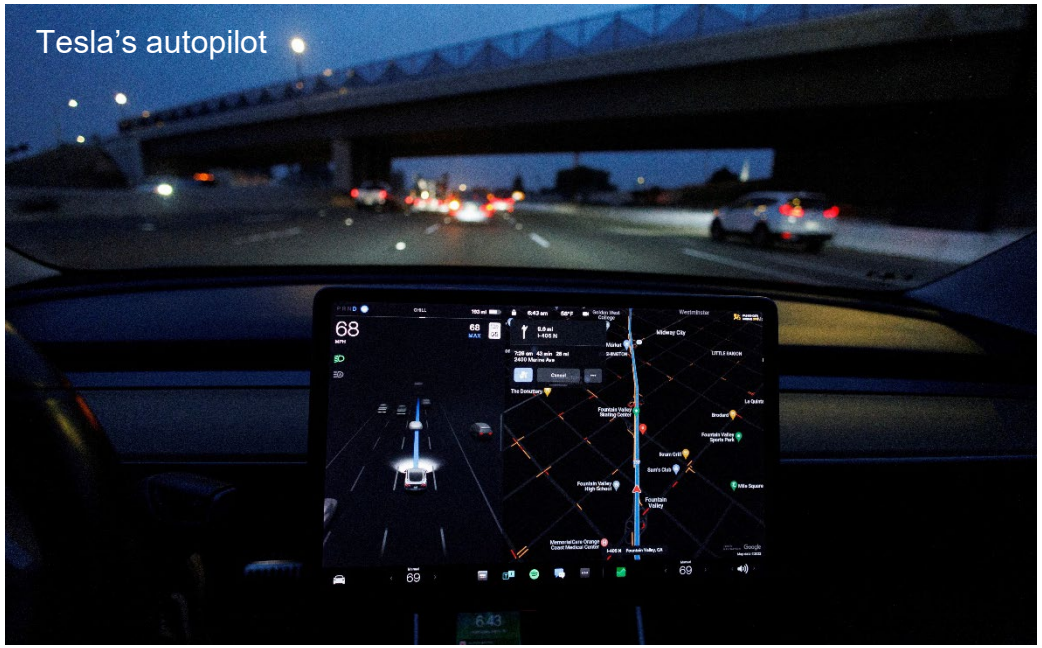
Robotic Mapping & Localization

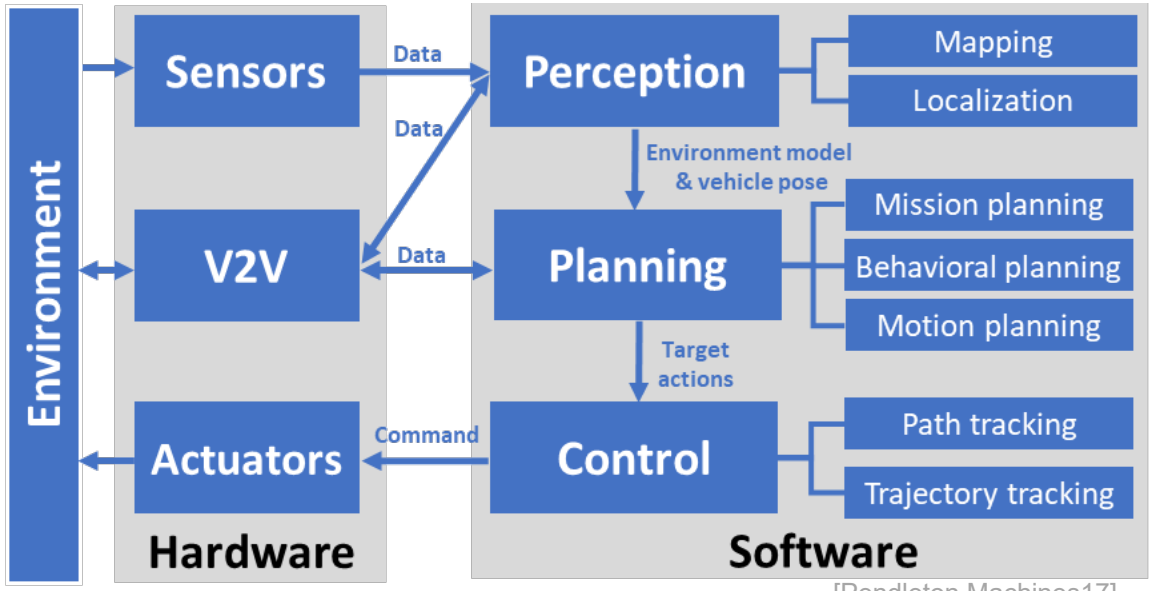
Kaveh Fathian
Assistant Professor
Computer Science Department
Colorado School of Mines

Lec16: Recap

Why We Need SLAM?

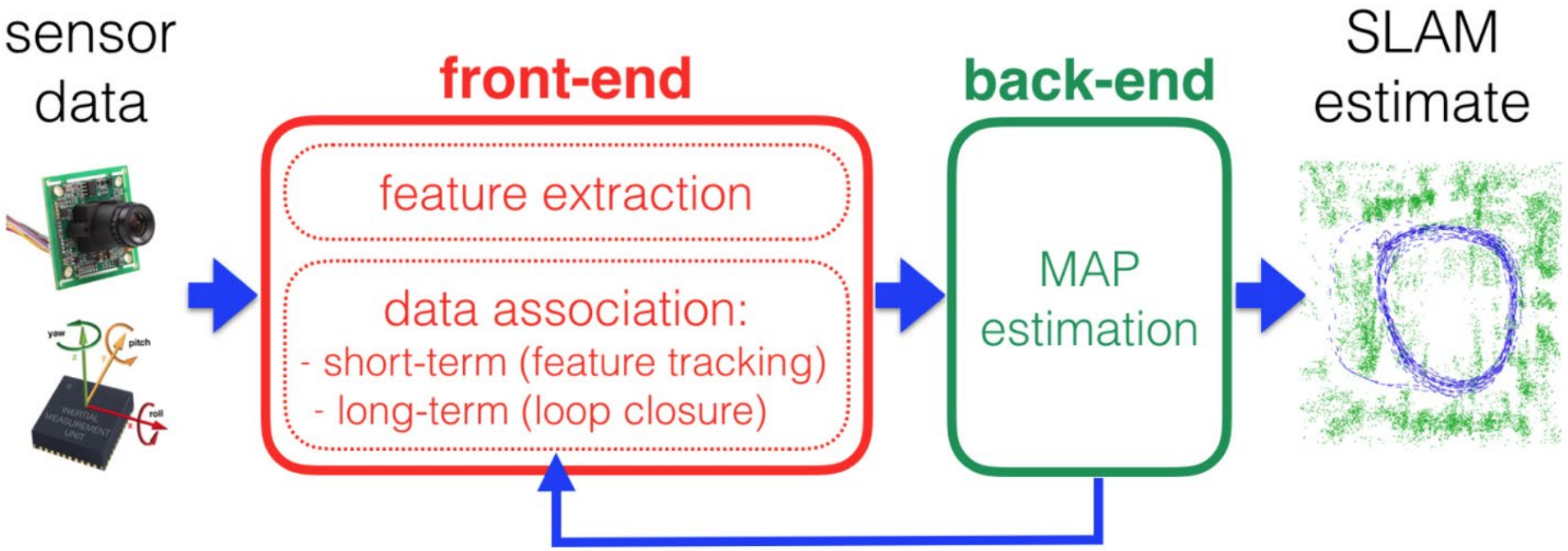
- **Autonomy** needs perception, planning, control
- **Perception**/SLAM ➔ understanding of environment/surrounding





[Pendleton Machines17]

SLAM Components



[Carlone, Visual Nav, MIT]

Standard SLAM pipeline consists of

- **Sensors**
- **Front-end** algorithms (real-time/online processing of sensor data, trajectory, map)
- **Back-end** algorithms (optimization/refinement of map, trajectory)

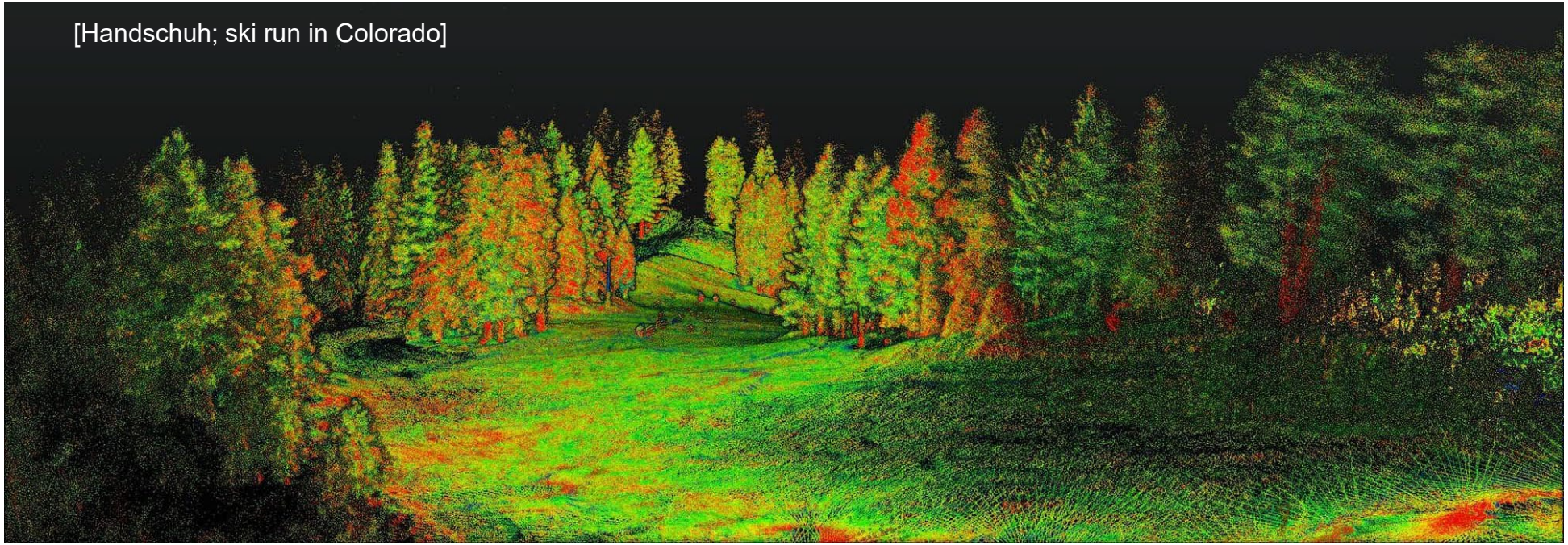
Mapping

- **Mapping**: is the process of creating a map of unknown environment
- **Representation**: is the format of the map
  - Point clouds
  - Polygons/patches
  - Occupancy grid
  - Semantic/objects
  - Metric vs. topological

[Lu, IEEE Access19]

Sparse

Dense

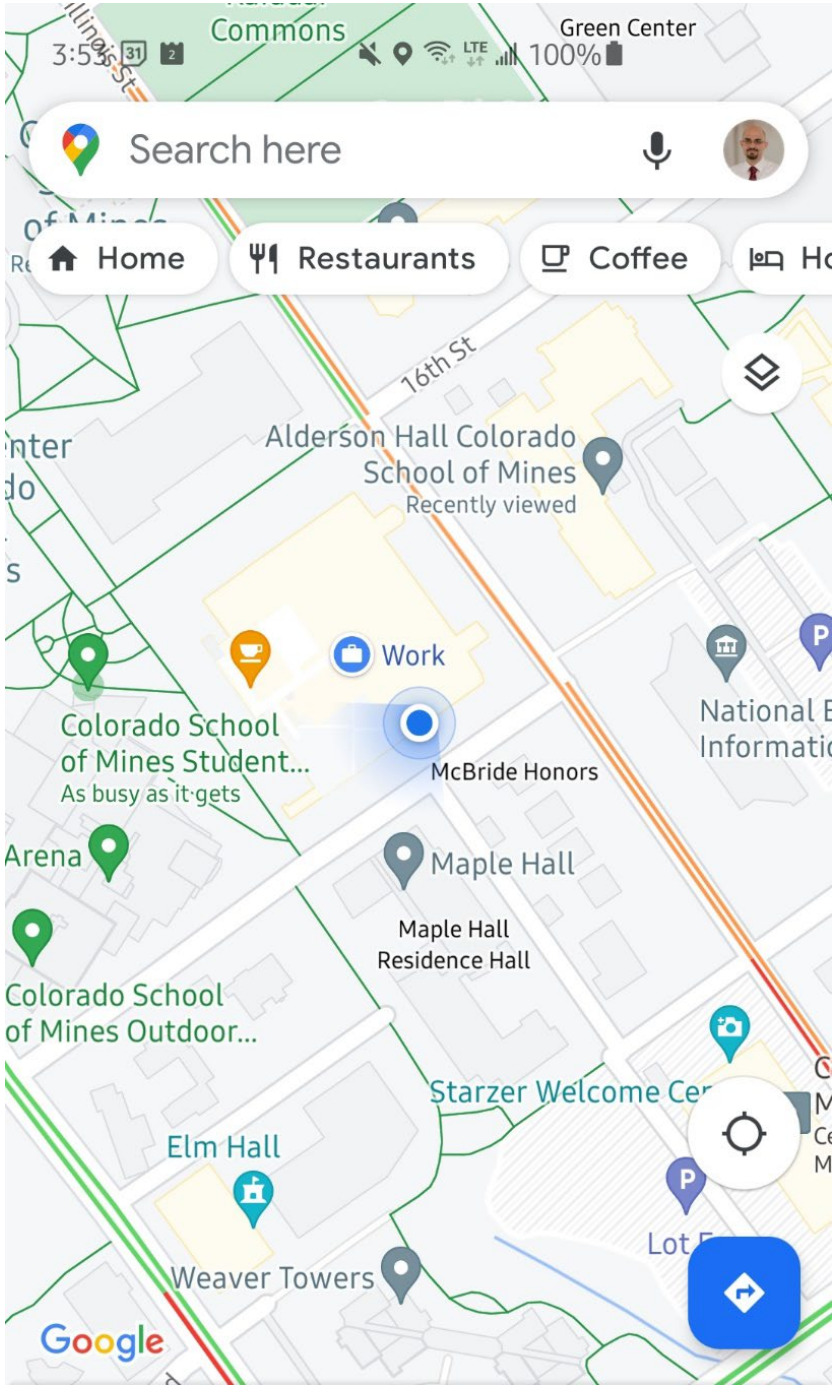


Localization

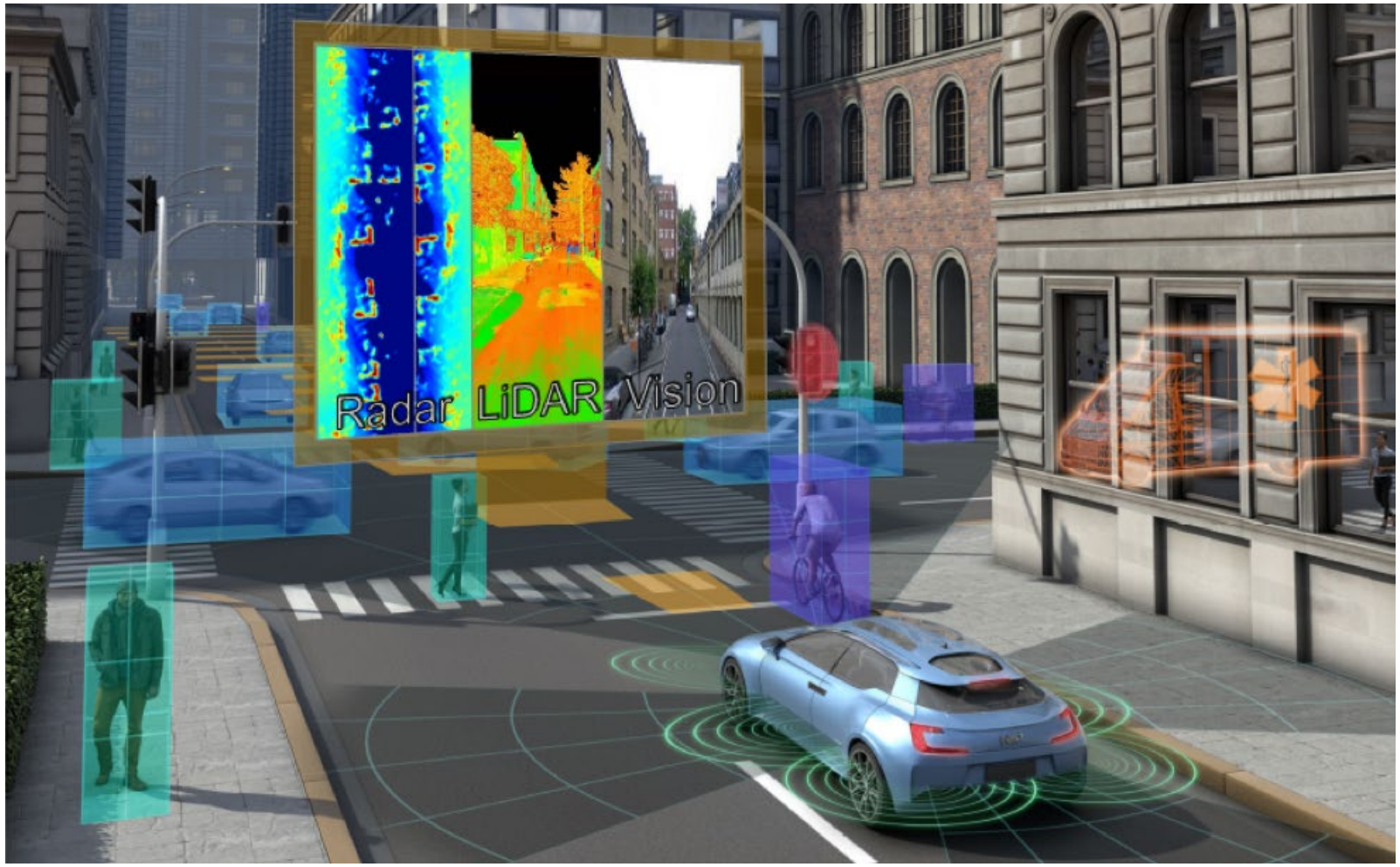
- **Localization**: is the process of determining precise **position** & **orientation** (aka **pose**) within environment

- Localization can be
  - In generated map
  - In given map—kidnapped robot problem!

- **Localization techniques**:
  - Particle Filters
  - Kalman Filters
  - Graph Optimization

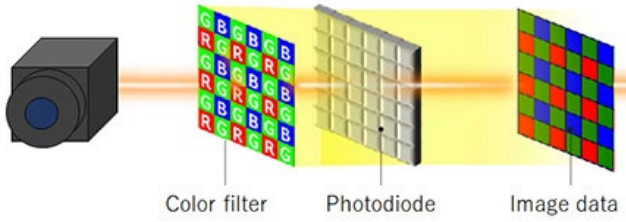


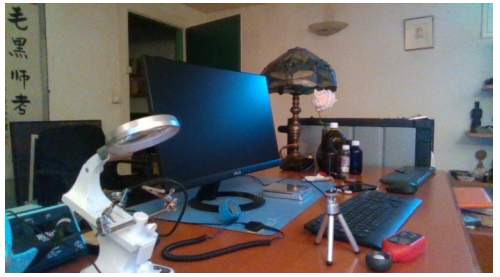
Sensors

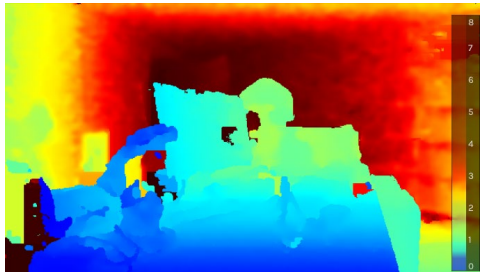


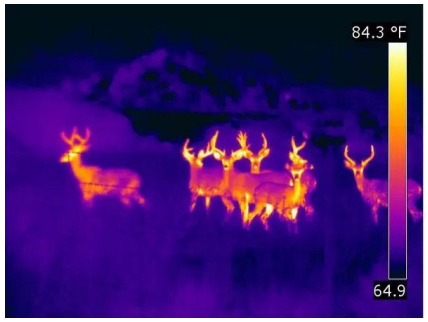
Sensors

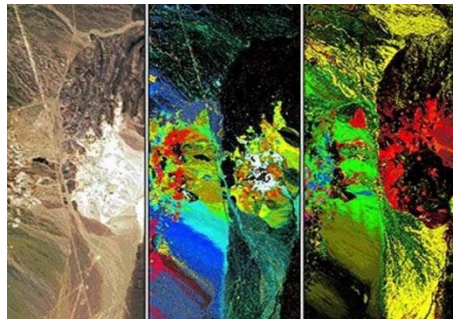
- Various sensors are used for SLAM/autonomy:
  - Camera
    - RGB
    - Stereo
    - Thermal/IR
    - Event
    - Hyperspectral
  - IMU
  - Encoder
  - LiDAR
  - Radar
  - Sonar
  - GPS

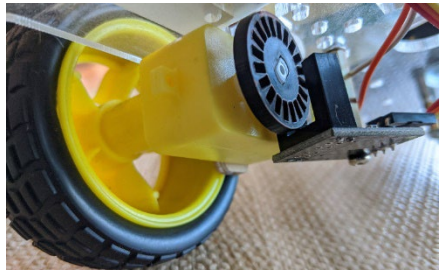


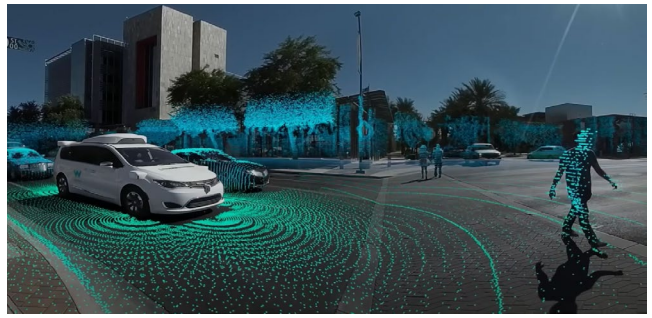




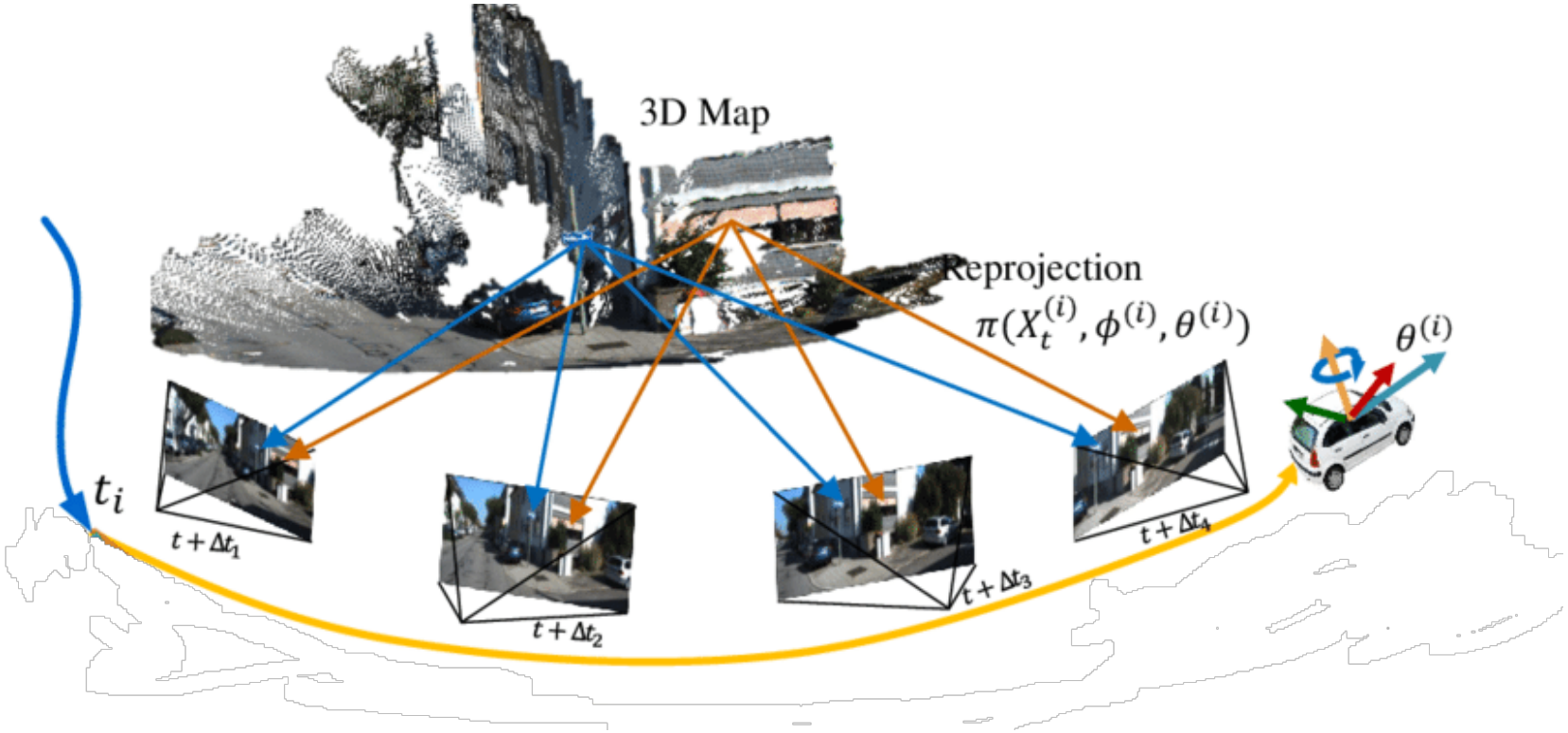




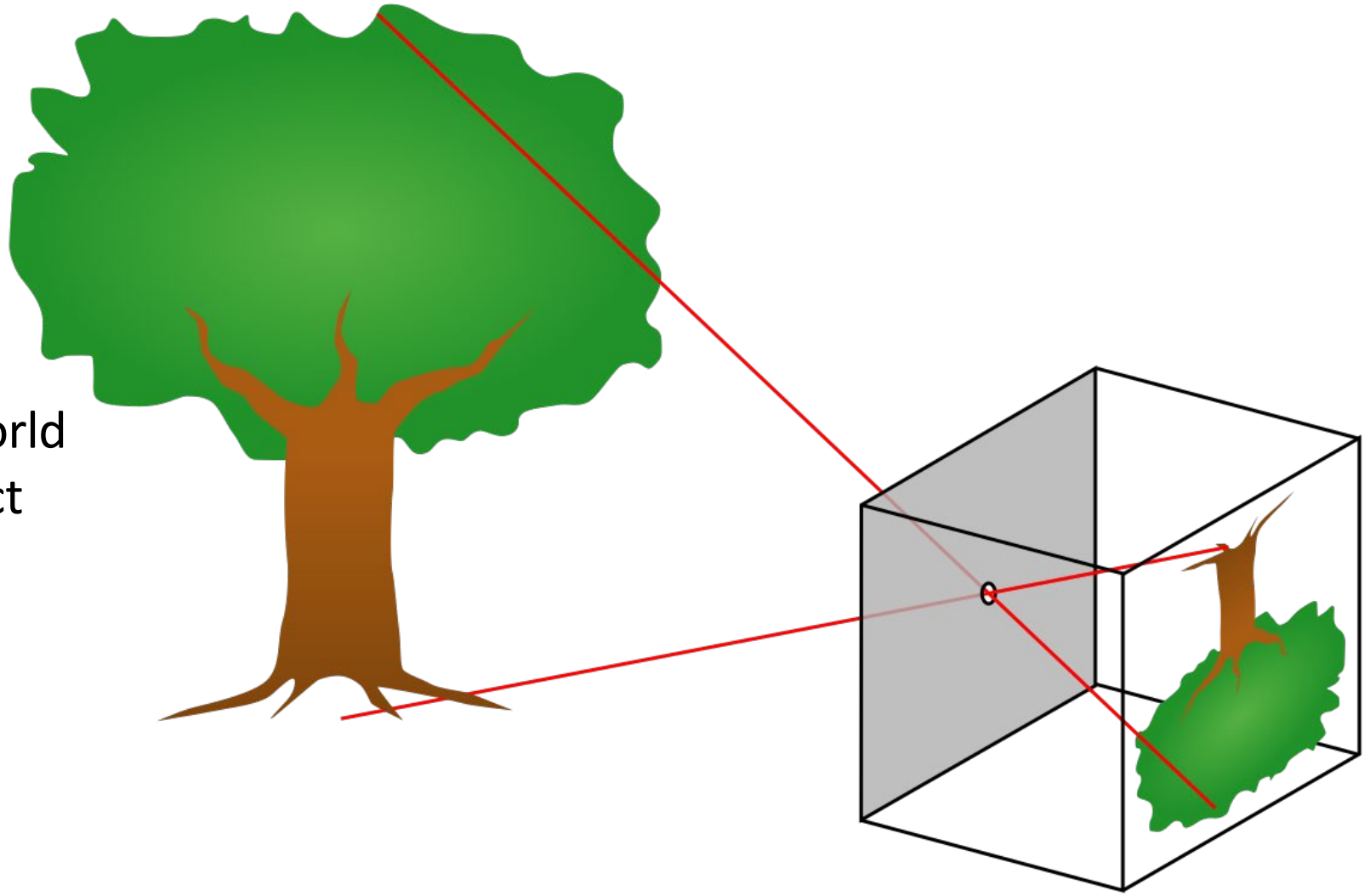




Visual Odometry



Pinhole Camera Model



The camera as a coordinate transformation

$$\mathbf{x} = \mathbf{P}\mathbf{X}$$

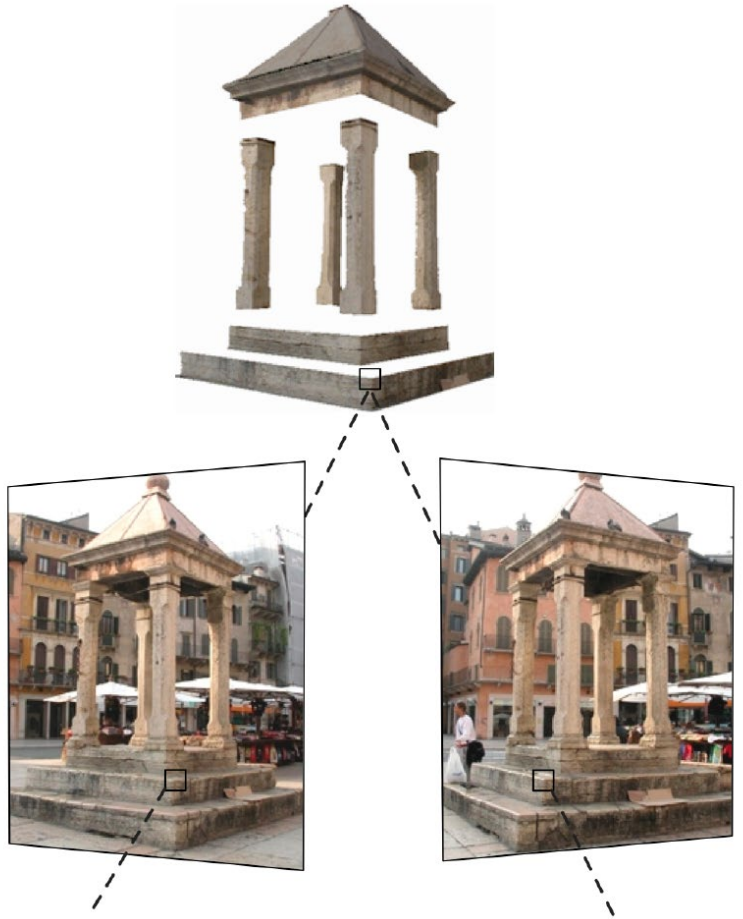
$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

homogeneous image coordinates 3 x 1

camera matrix 3 x 4

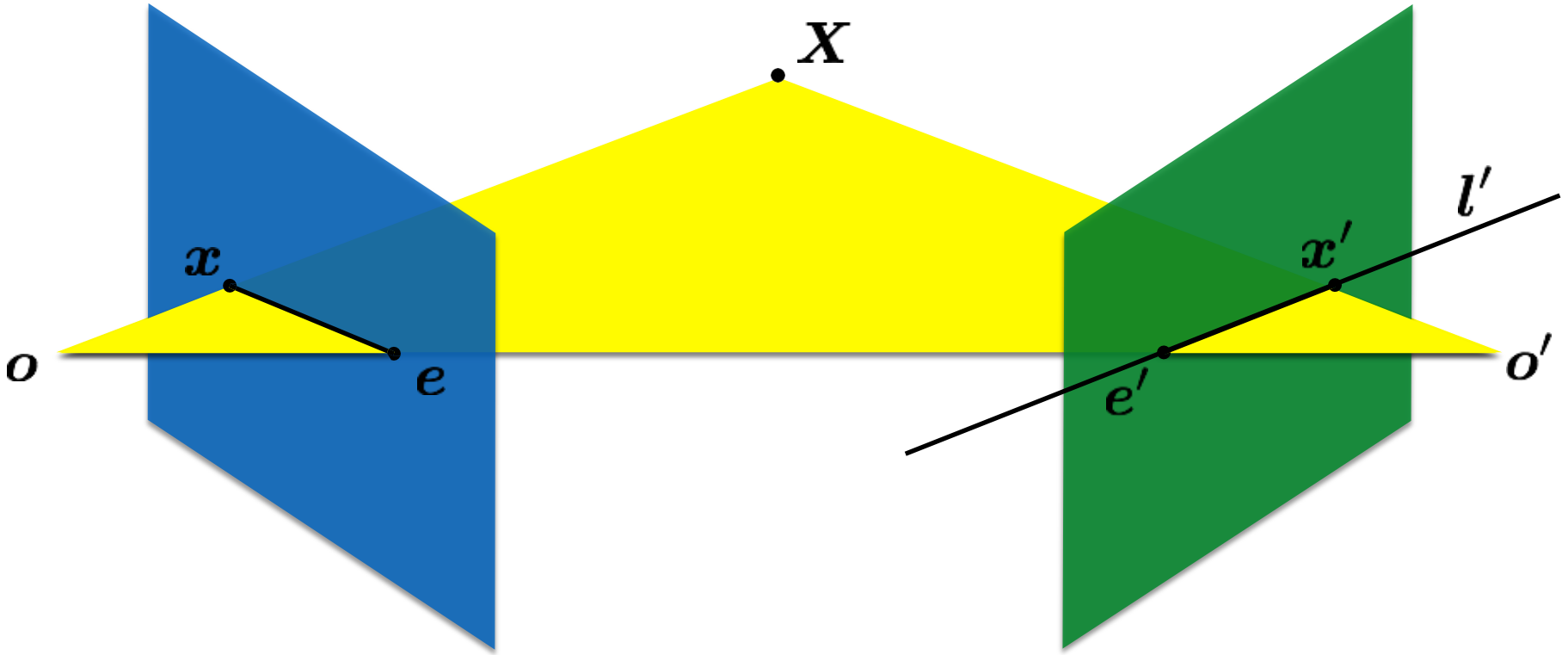
homogeneous world coordinates 4 x 1

Two-View Geometry



Essential Matrix

$$\boldsymbol{x}'^{\top}\mathbf{E}\boldsymbol{x} = 0$$



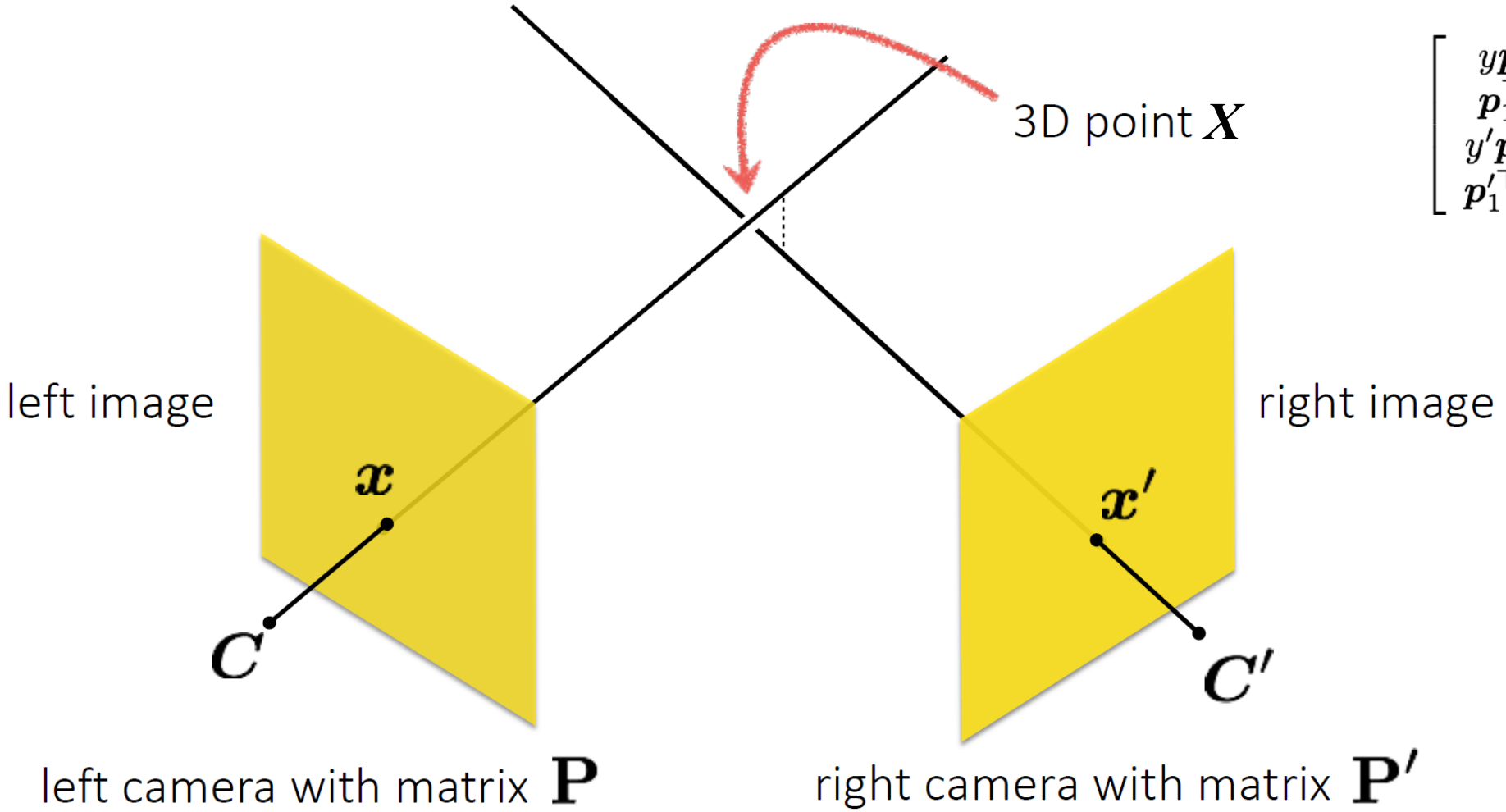
Homography Matrix

Given matched image points (P, P'), the homography matrix H relates

$$P' = H \cdot P$$

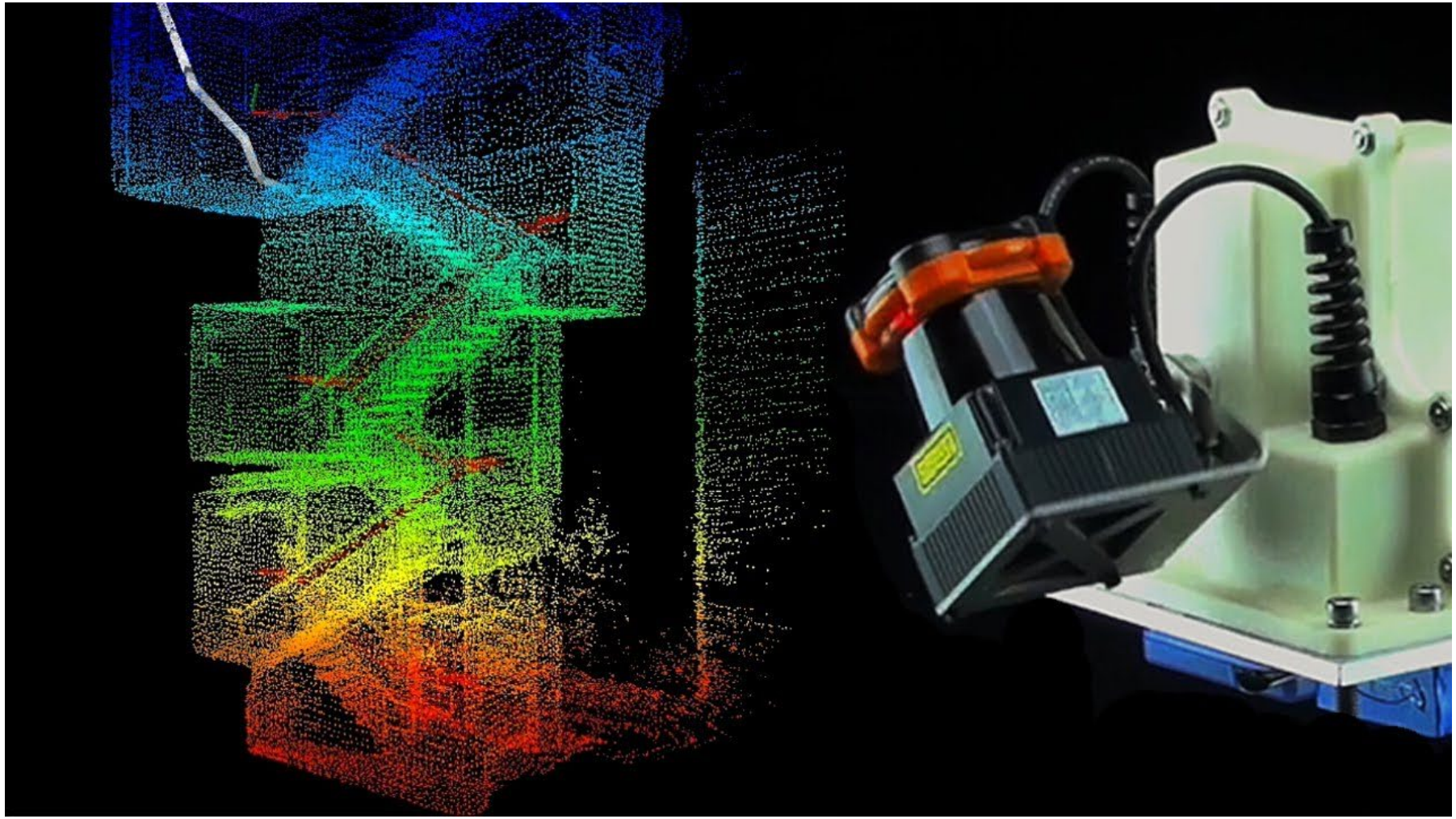
Note: Homography holds only for co-planar points or rotation-only motion

Triangulation



$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \\ y'\boldsymbol{p}_3'^\top - \boldsymbol{p}_2'^\top \\ \boldsymbol{p}_1'^\top - x'\boldsymbol{p}_3'^\top \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

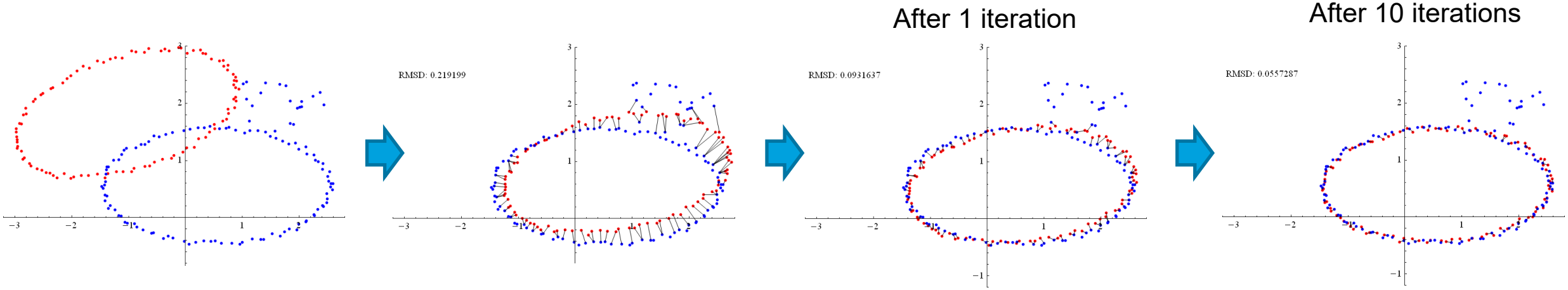
LiDAR Odometry



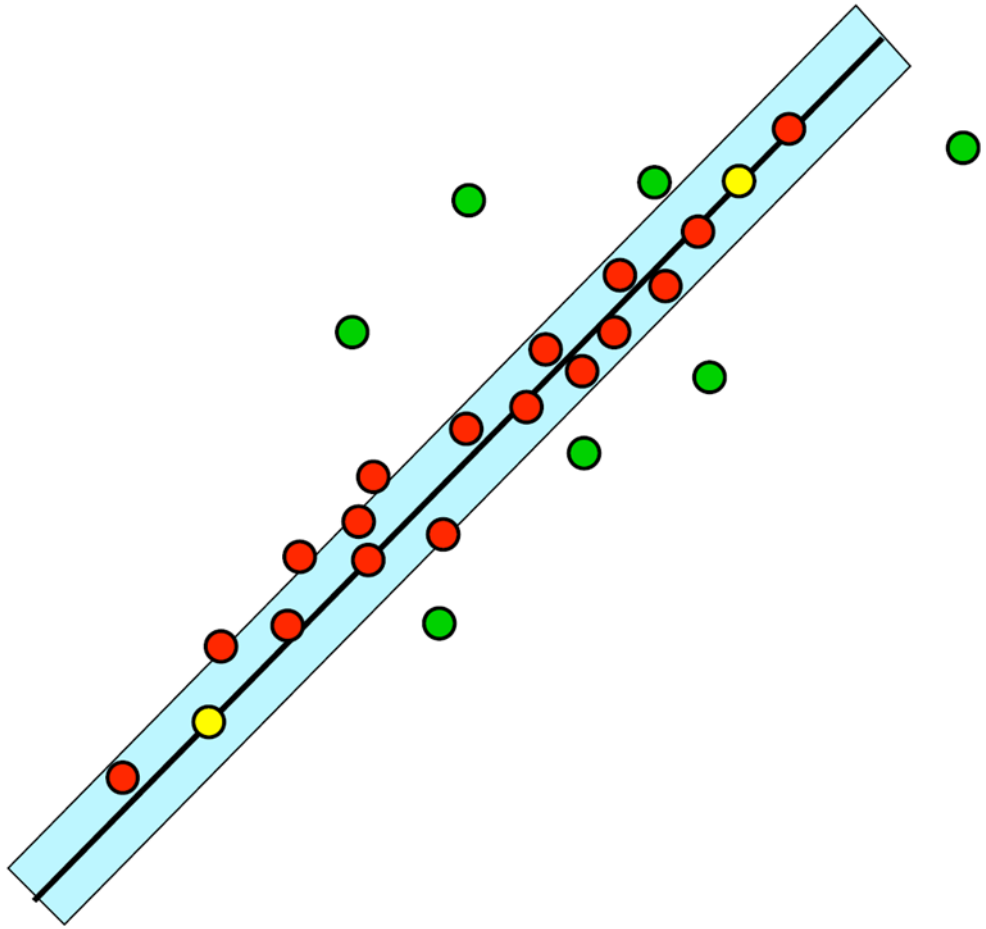
Iterative Closest Point (ICP) algorithm - overview

- **Step 1:** Determine associations, $\mathcal{I}$, between target and source by finding the nearest neighbor.

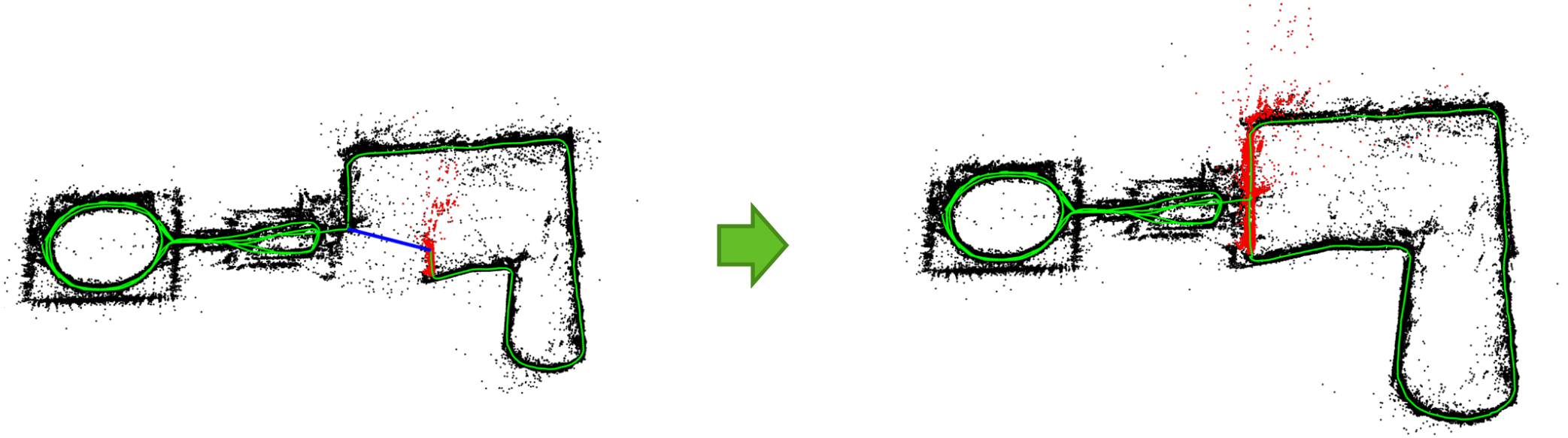
- **Step 2:** Given those associations, minimize the residual between points.



RANSAC

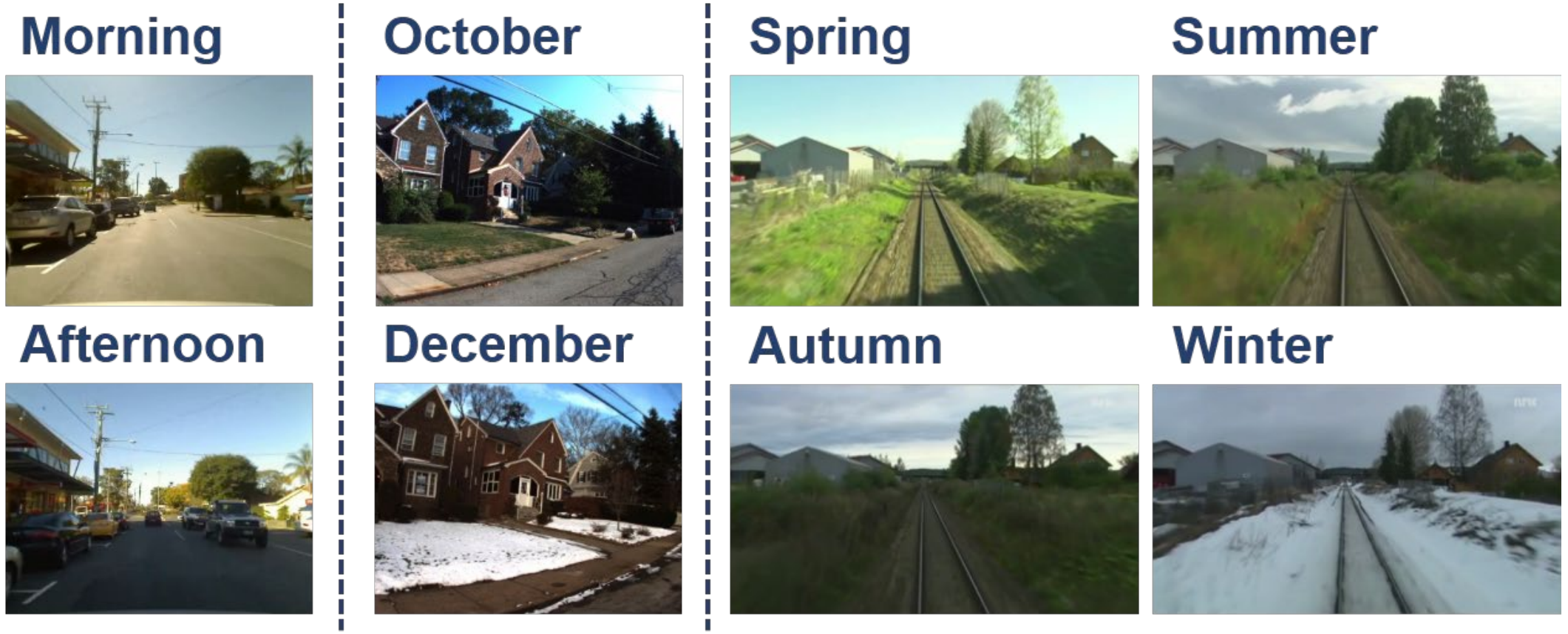


Loop Closure



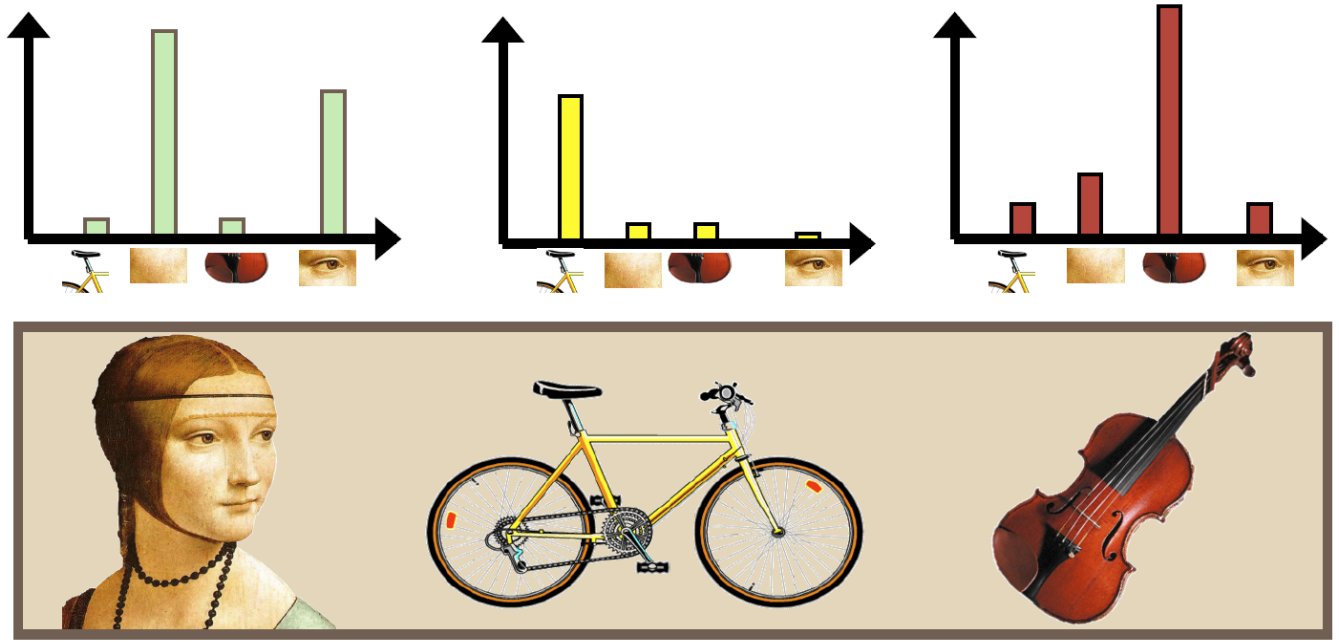
Place Recognition

- Place recognition challenges:
  - Time of day; weather; viewpoint; dynamic environment, seasons, etc.
- Wrong loop closure is catastrophic!

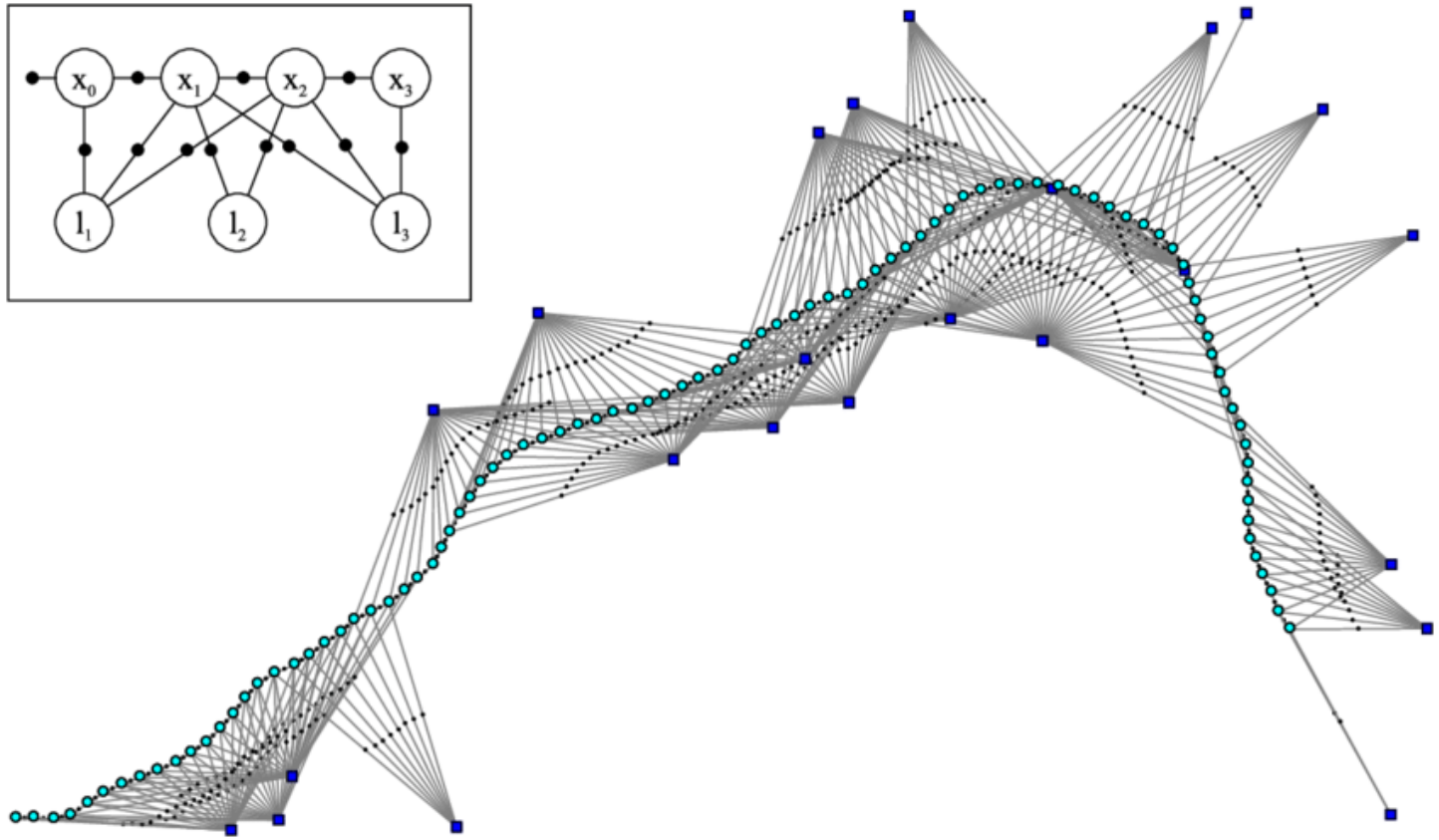


Bag of Visual Words

- Visual words = independent features
- Words from the dictionary
- Represent the images based on a histogram of word occurrences



Factor Graph Optimization



Geometric Consistency

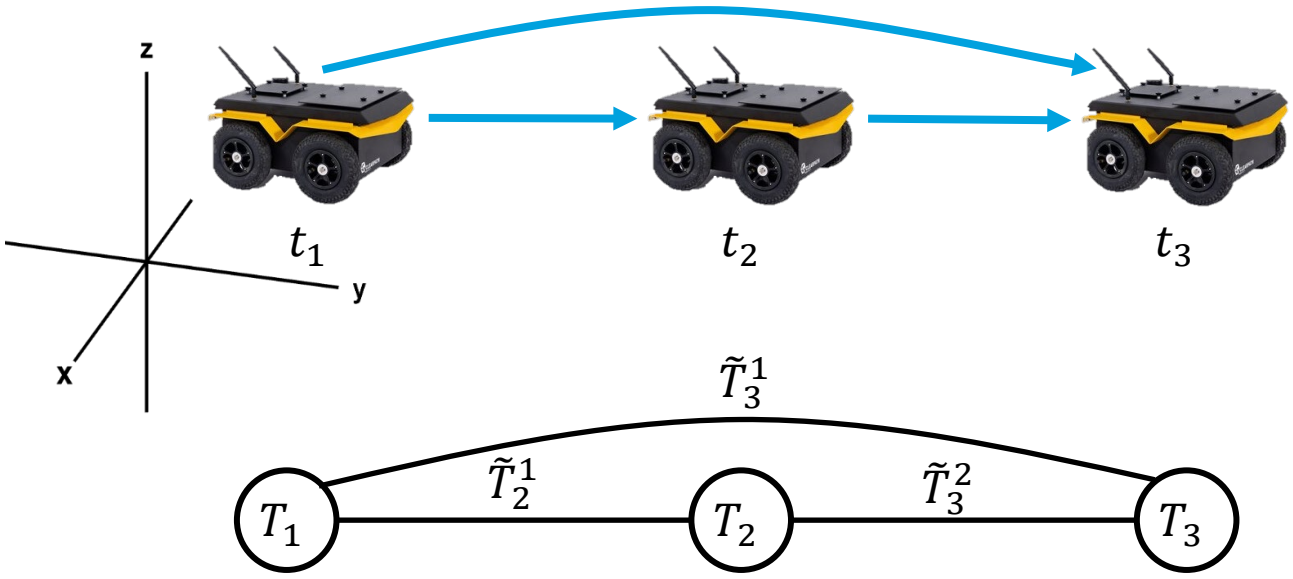
Example: Suppose we compute odometry estimates between t_1, t_2, t_3

- Estimates can be geometrically inconsistent (always are in practice!)
- We can take "**average**" of the estimates to obtain consistent results

Factor Graph Optimization

Factor graph: Mathematical framework for solving estimation/inference problems & obtain consistency

- Problem representation as a graph
- **Nodes/Vertices**: are optimization variables. Can be of different types (continuous or discrete, poses or landmarks, etc.)
- **Factors/Edges**: represent relationship/constraints between variables (e.g. relative pose estimates, sensor measurements)



Pose/factor graph optimization:

$$\min_{T_t \in \mathrm{SE}(3)} \sum_{(i,j)\in\varepsilon} \left\| T_i^{-1} T_j - \tilde{T}_{ij} \right\|_{\Sigma_{ij}}^2$$

Solvers:

- GTSAM (https://gtsam.org/)
- G2o (https://openslam-org.github.io/g2o.html)
- Ceres (http://ceres-solver.org/)

Research Opportunities

Monocular SLAM Study

We conducted a study on SOTA mono-SLAM algorithms to

- ❑ Identify specific weaknesses in mono-SLAM systems
- ❑ Gain insights into open problems in the field

9 SOTA SLAM (and SfM) systems:

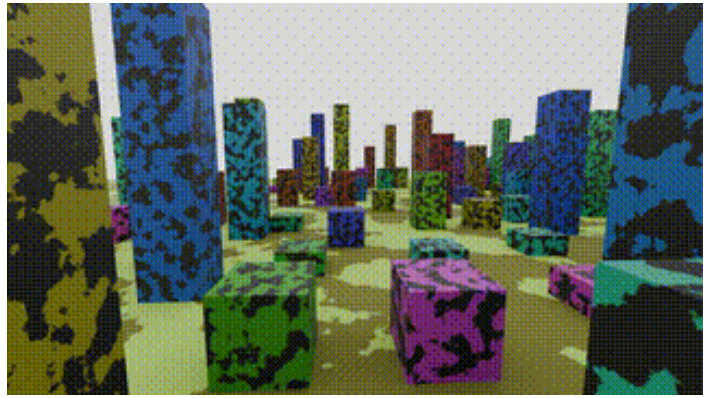
- ❑ ORB-SLAM3 (https://github.com/UZ-SLAMLab/ORB_SLAM3)
- ❑ PLP (https://github.com/PeterFWS/Structure-PLP-SLAM)
- ❑ SVO (https://github.com/uzh-rpg/rpg_svo_pro_open)
- ❑ LSD (https://github.com/tum-vision/lsd_slam)
- ❑ DSO (https://github.com/JakobEngel/dso)
- ❑ DROID (https://github.com/princeton-vl/DROID-SLAM)
- ❑ DPV-SLAM (https://github.com/princeton-vl/DPVO)
- ❑ COLMAP (https://github.com/colmap/colmap)
- ❑ GLOMAP (https://github.com/colmap/glomap)

7 major datasets used for testing

- ❑ EuRoC (http://projects.asl.ethz.ch/datasets/doku.php)
- ❑ TUM (https://vision.in.tum.de/data/datasets/visual-inertial-dataset)
- ❑ KITTI (https://www.cvlibs.net/datasets/kitti/eval_odometry.php)
- ❑ Newer College (https://ori-drs.github.io/newer-college-dataset/)
- ❑ UZH FPV (https://fpv.ifi.uzh.ch/datasets/)
- ❑ TartanAir (https://theairlab.org/tartanair-dataset/)
- ❑ Our novel synthetic dataset (ready for open-source release)

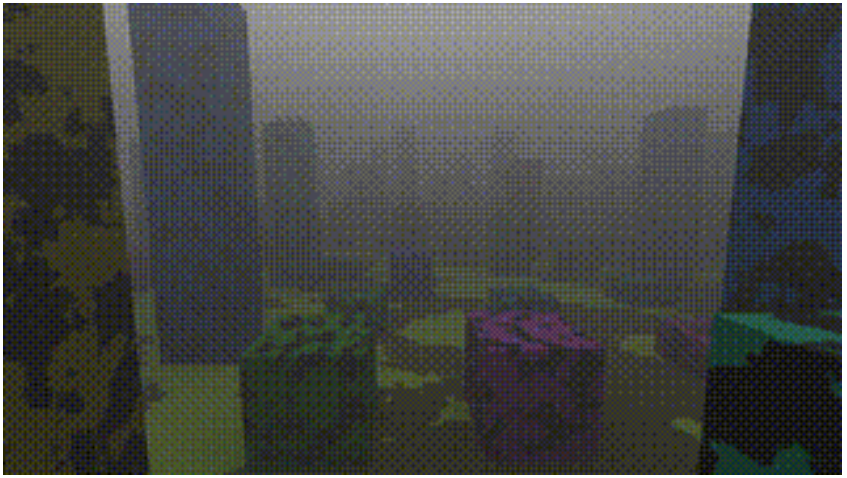
Monocular SLAM Study

- ❑ We created a synthetic “block world” in Blender
- ❑ Simulated a camera on a flying drone
- ❑ Baseline video represents perfect conditions for a monocular SLAM system

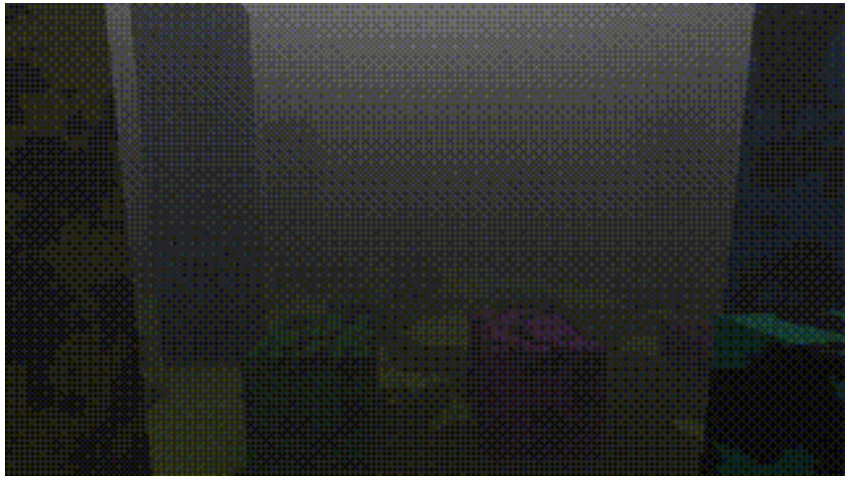


Monocular SLAM Study

Fog



Easy

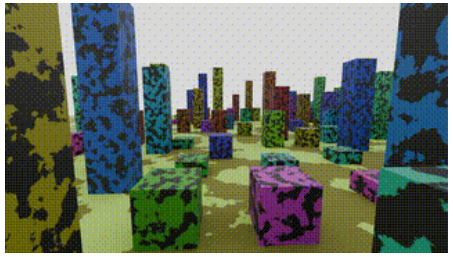


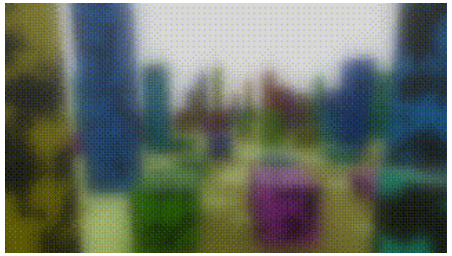
Medium

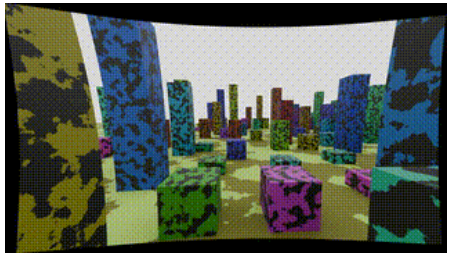
Hard

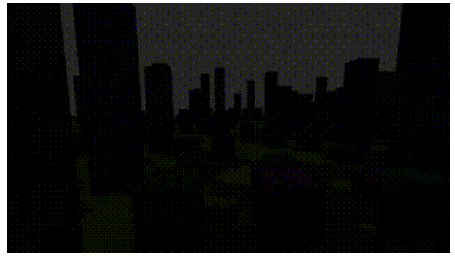
- ❑ We created 27 challenging sequences to understand how systems handle them
- ❑ Each challenge has three levels of difficulty
- ❑ Other than the variable being modified by the challenge, the scene is identical

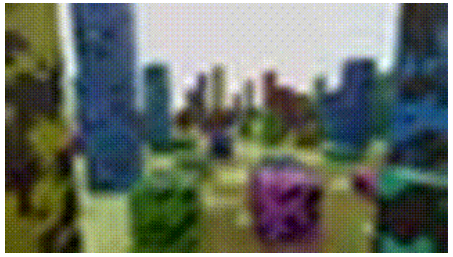
More Examples

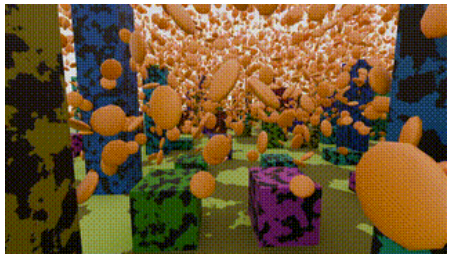

Baseline

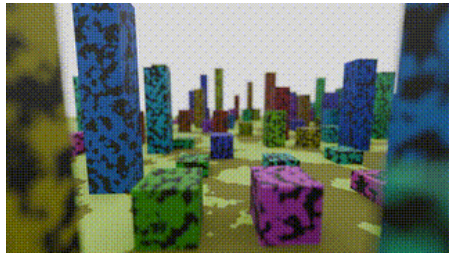

Blur (easy/med/hard)

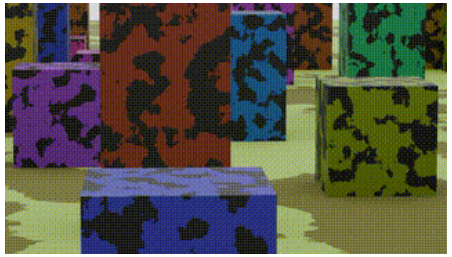

Calibration (easy/med/hard)

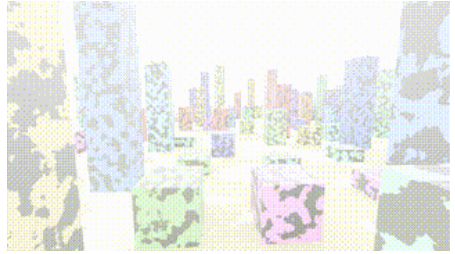

Night (easy/med/hard)

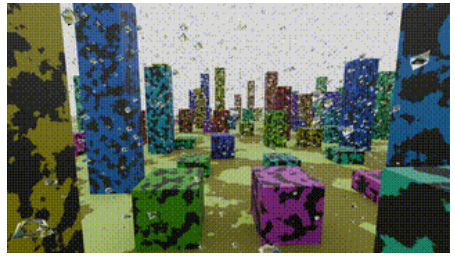

Resolution (easy/med/hard)

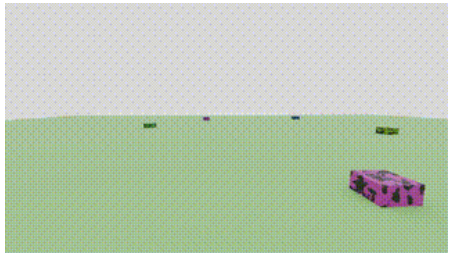

Leaves (easy/med/hard)

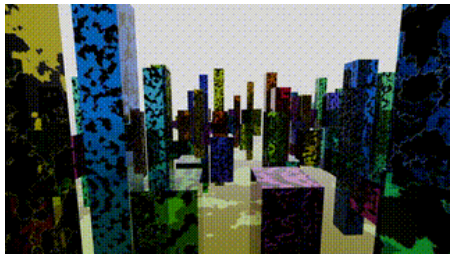

Depth-of-Field (easy/med/hard)

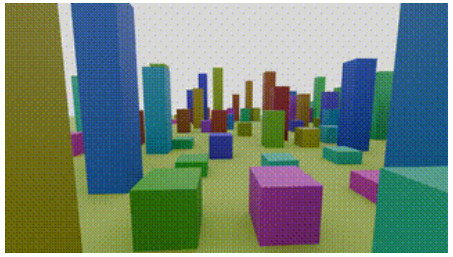

Narrow FoV(easy/medium/hard)

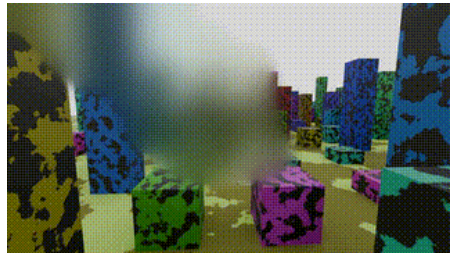

Exposure (easy/med/hard)

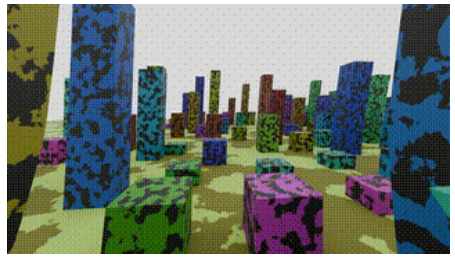

Rain (easy/med/hard)


Featureless (easy/med/hard)


Reflective (easy/med/hard)

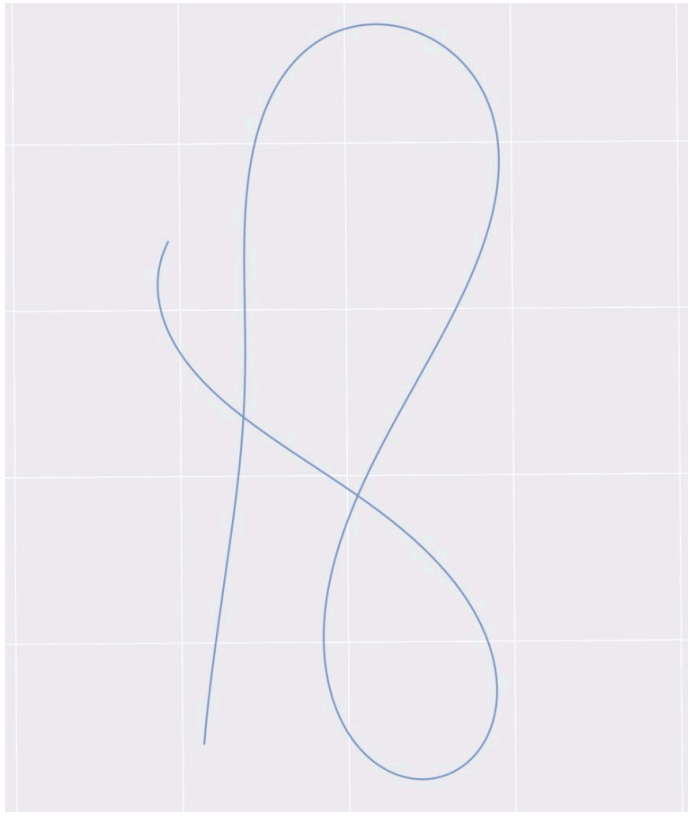

Texture (easy/med/hard)


Smear (easy/med/hard)

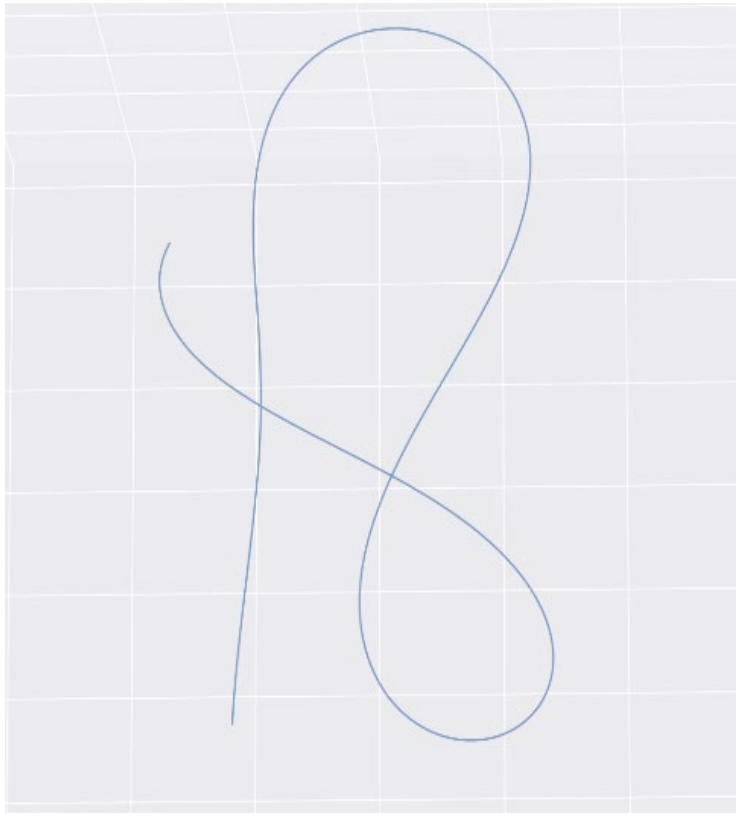

Rolling Shutter (easy/med/hard)

Evaluation

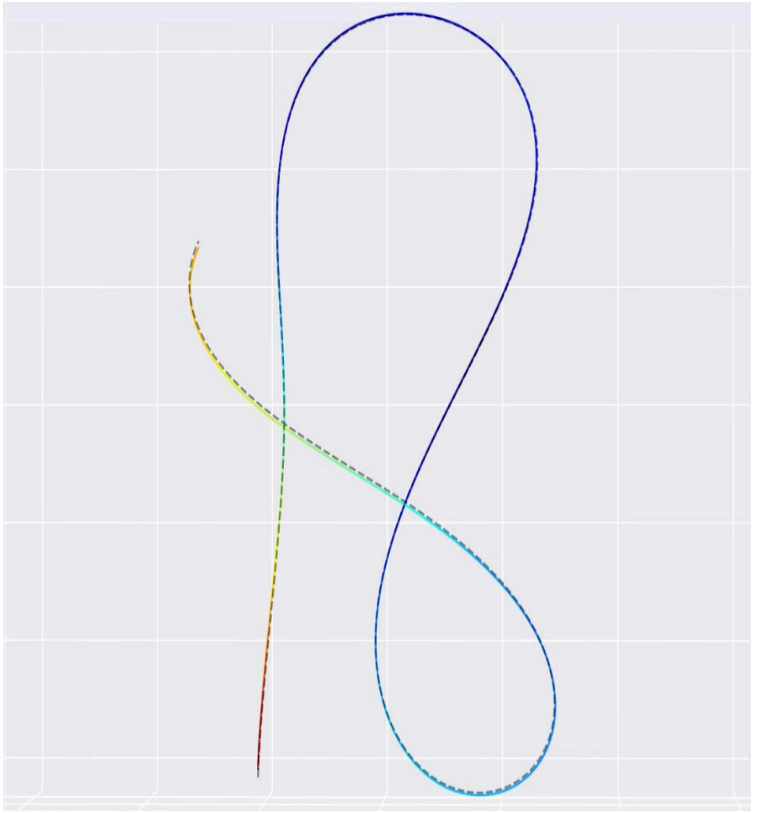
- ❑ We run every system on every sequence 50 times
- ❑ Evaluation metric is Absolute Trajectory Error (ATE)
- ❑ ATE compares the distance between each point on the estimated trajectory with where it should be using the ground truth



Ground Truth

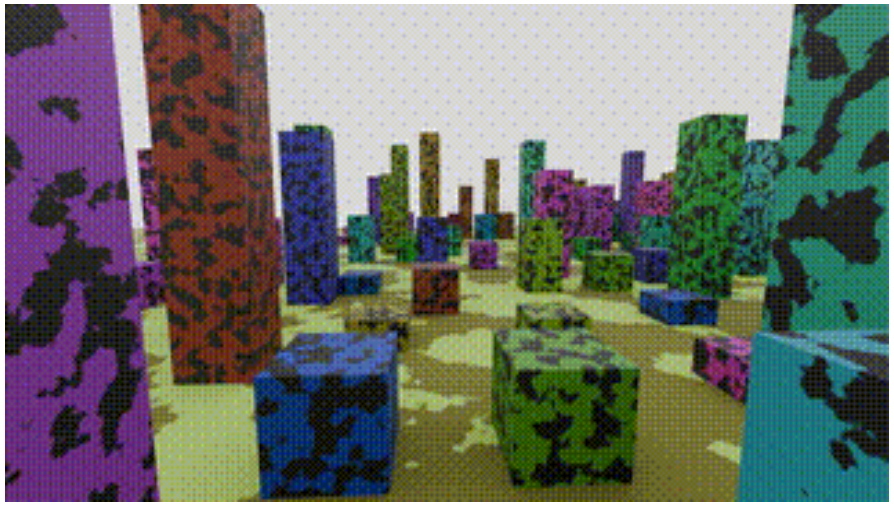


Trajectory Estimated by DROID-SLAM

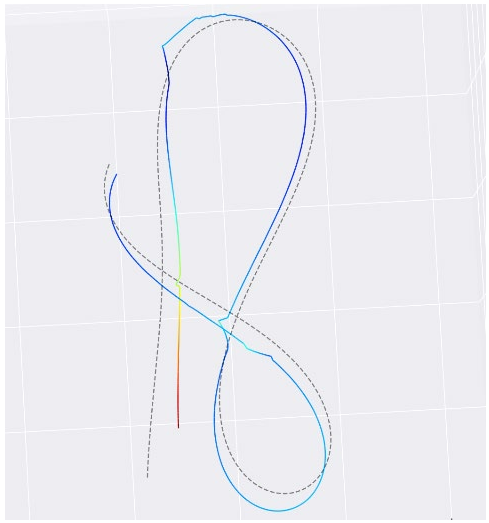


ATE RMSE: 2.8cm

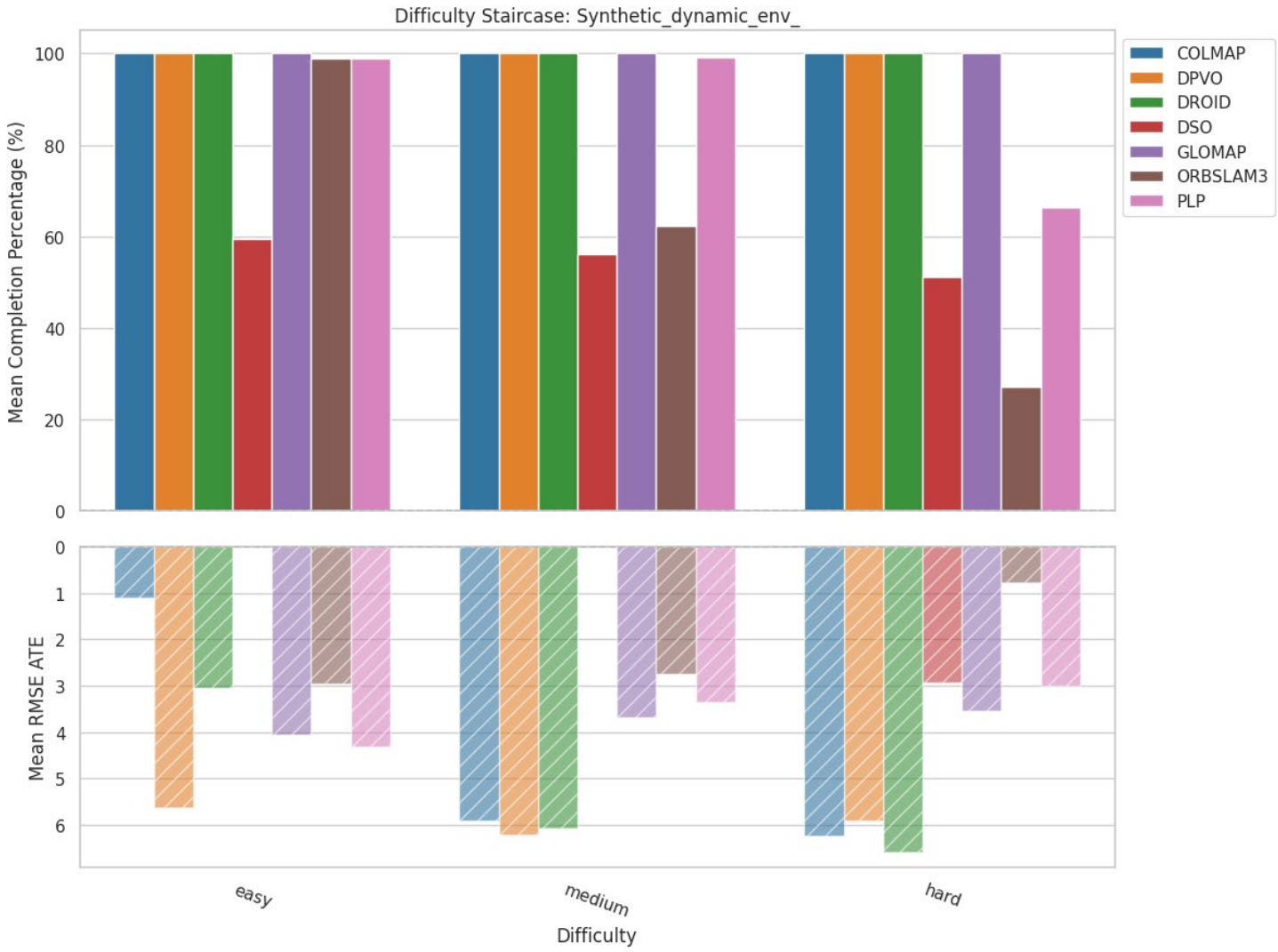
Insights



Dynamic Env Easy

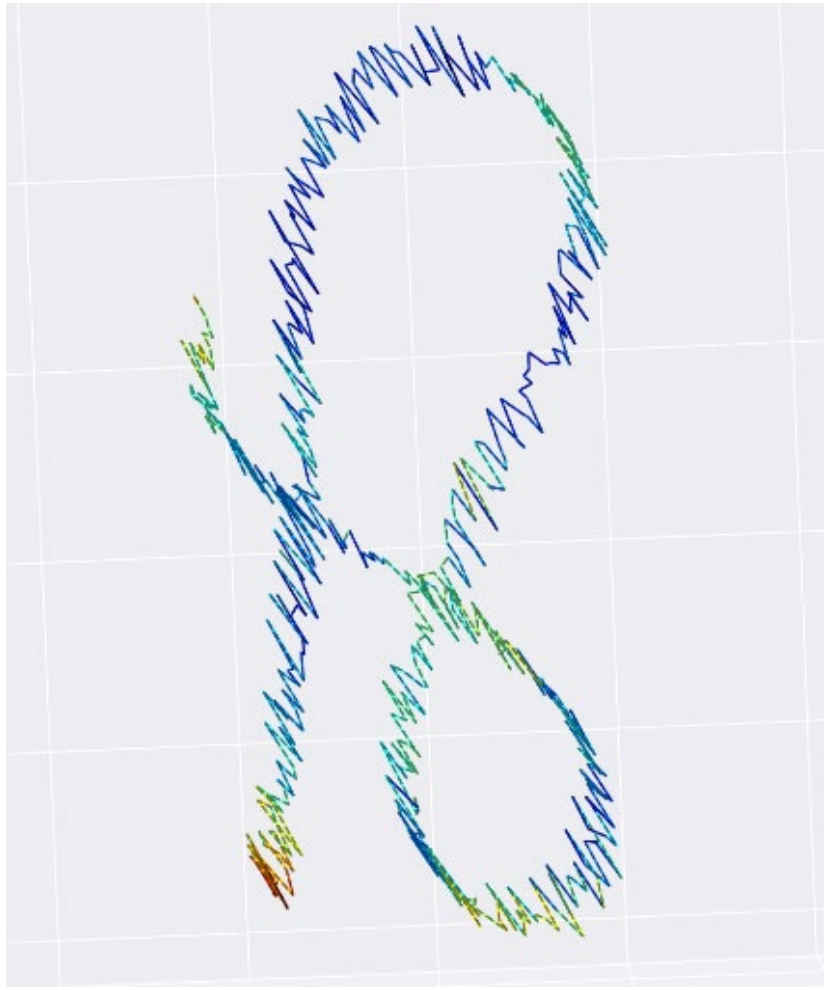


COLMAP 91x real time

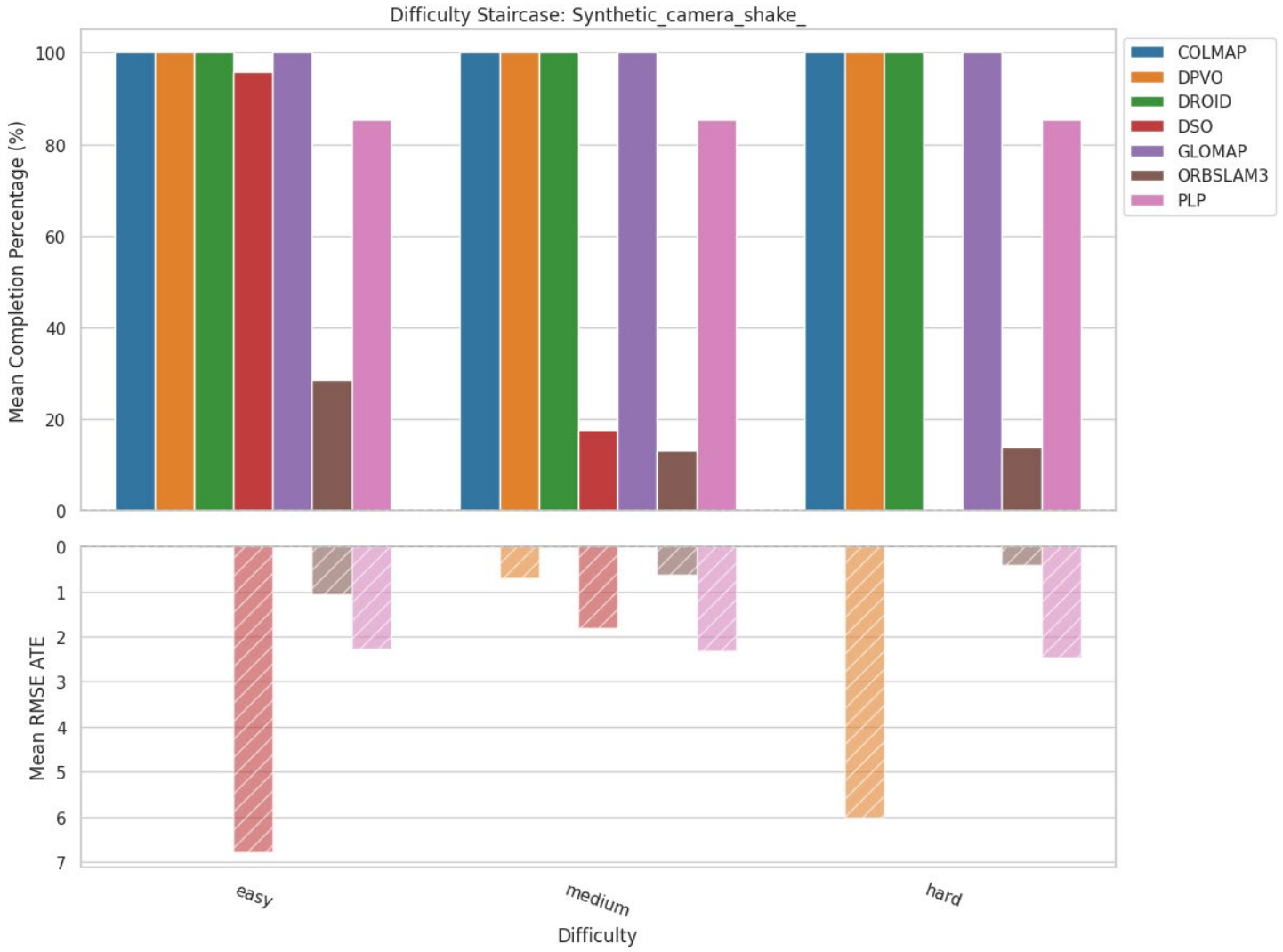


Nobody can handle dynamic environments!

Insights

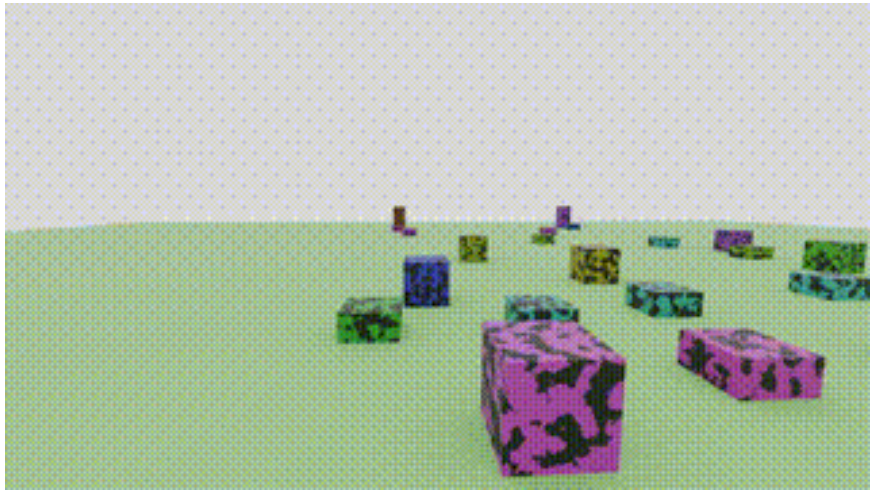


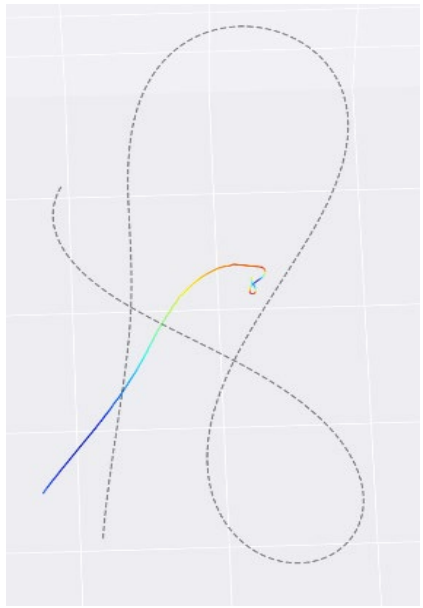
DROID 12.3x real time



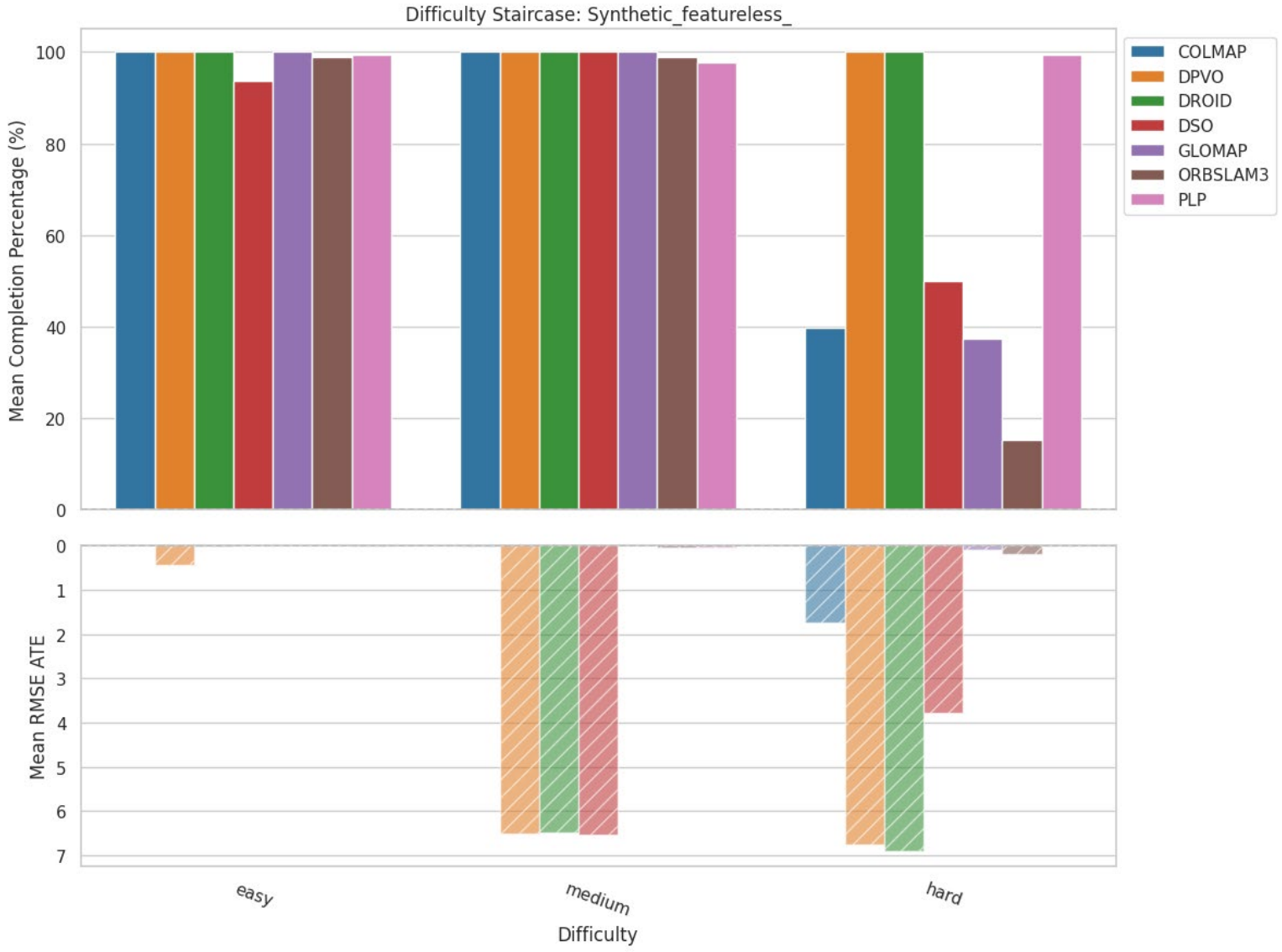
DL methods can be weirdly robust to camera shake

Insights





DROID 3x real-time



… but weirdly bad at featureless environments

Conclusion

- ❑ SLAM is an old technology, however, may open/difficult problems remain
- ❑ SOTA SLAM systems are not reliable in challenging conditions
- ❑ Machine learning (DL) has enabled unprecedented capabilities, and many capabilities are yet to be discovered as ML technology evolves (e.g., LLMs)

SLAM Research Opportunities

We Pioneer SLAM Algorithms for Challenging Env

- Summer & Fall semester openings
- Teams on SLAM, autonomy, hardware
- Apply at www.ariarobotics.com/join-us

www.ariarobotics.com