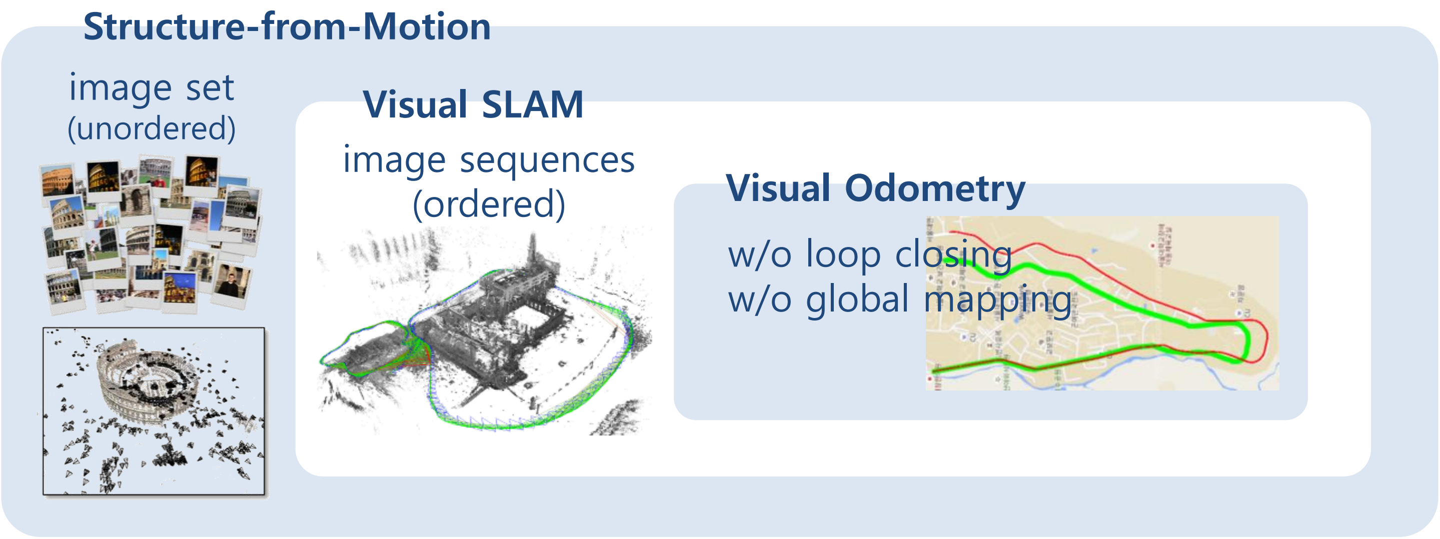
Note) Slides and example codes are available:
https://github.com/mint-lab/3dv_tutorial

An Invitation to 3D Vision: Visual SLAM and Odometry

Sunglok Choi, Assistant Professor, Ph.D.
Dept. of Computer Science and Engineering, SEOULTECH
sunglok@seoultech.ac.kr | https://mint-lab.github.io/

SfM and Visual SLAM/Odometry

- **Structure-from-Motion (SfM)** → 3D Reconstruction, Photo Browsing
  - Bundler, COLMAP, MVE, Theia, openMVG, OpenSfM, Toy SfM / VisualSFM (GUI, binary only)
- **Visual SLAM** → Augmented Reality, Navigation (Mapping and Localization)
  - PTAM (Parallel Tracking and Mapping), DTAM (Dense Tracking and Mapping), ORB-SLAM2, LSD-SLAM
  - cf. Visual loop closure (a.k.a. visual place recognition): DBoW2, FBoW, PoseNet, NetVLAD
- **Visual Odometry** → Navigation (Localization)
  - LIBVISO2 (C++ Library for Visual Odometry 2), SVO (Semi-direct Monocular Visual Odometry), DVO (Direct Sparse Odometry), DeepVO, UnDeepVO



Overview

- **SLAM**: Joint estimation of robot poses (or path) and a map (used in localization)
  - A chicken-and-egg problem
  - One of the most popular topics in robot navigation (mobile robot)

- **Why SLAM?**
  - Autonomous navigation needs information about robot pose.
    - In indoor, GPS is not available.
    - In outdoor, GPS is not perfect and complete.
      - e.g. inaccurate (due to multi-path) and unavailable (due to urban canyons, tunnels, …)
    - Dead-reckoning with IMUs or encoders suffers from drift error.
  - Map-based localization (e.g. using landmark maps or HD maps)
    - If a robot starts to navigate on an unknown environment.
    - If the environment was changed. (e.g. new or removed landmarks)

Overview

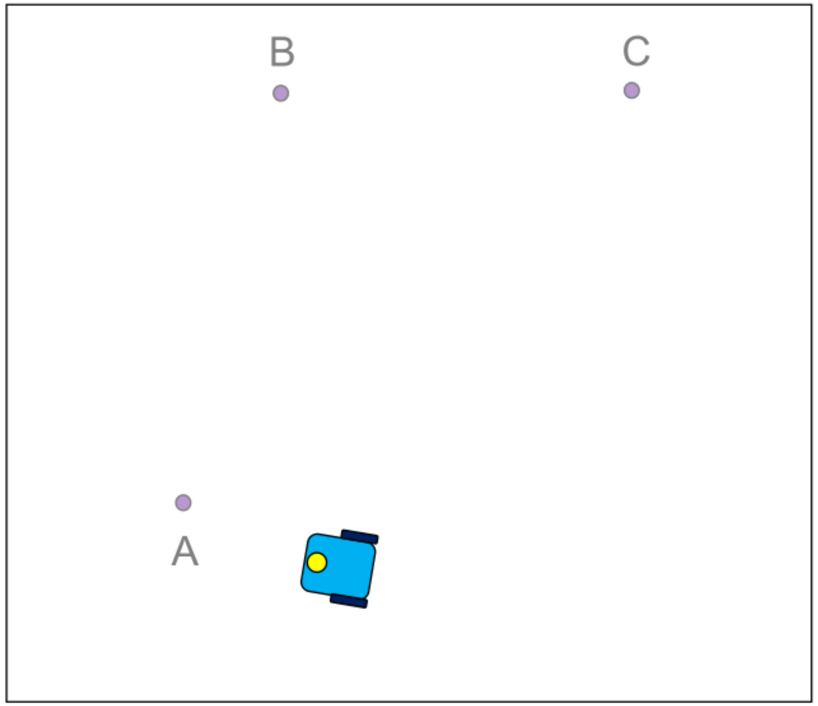
- **Why many variants?**
  - Sensor modalities: Camera, LiDAR, GPS, … / encoders, IMU, …
    - Data utilization: Feature-based (indirect) vs.. direct / sparse vs.. dense
  - Map representations: Feature maps vs.. metric maps, keyframe maps, topological maps (~ pose graphs), …
    - Dimension of robot pose and features (space): 2D vs.. 3D
  - Working scenarios: Indoor, on-road, underwater, flying (~ handheld, wearable), …

- **Applications**
  - Robot/vehicle navigation
  - Augmented/virtual reality
  - 3D capture and reconstruction
  - …

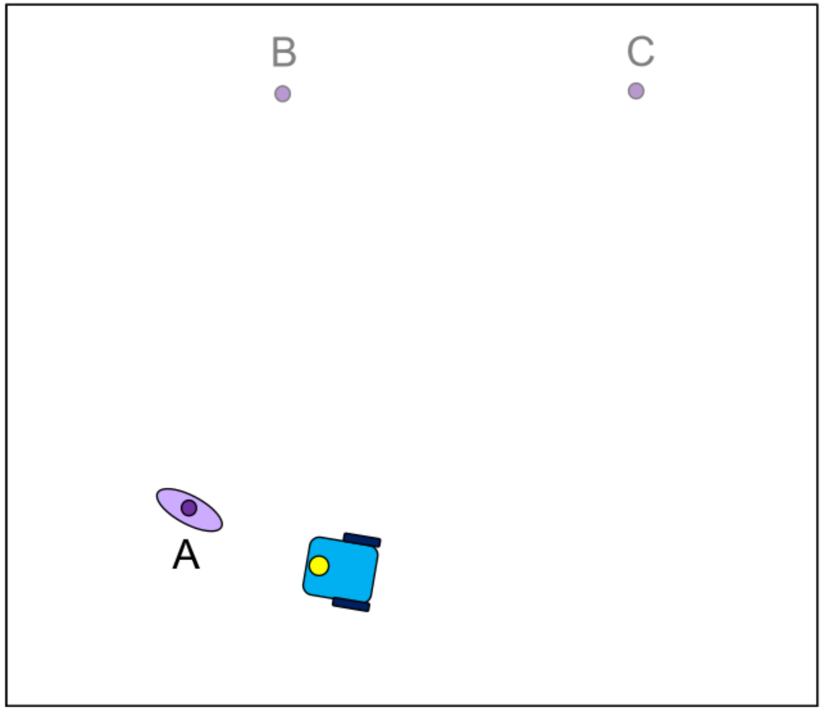
Useful References

- Books and papers (ordered by their difficulties)
  - SLAM Tutorial @ ICRA 2016 and @ RSS 2015
  - A Tutorial on Graph-based SLAM [ITSM, 2010]
  - SLAM Course by Cyrill Stachniss [2013-14]: Slides, YouTube
  - ~~SLAM Summer School 2006 and SLAM Part I and II [RAM, 2006]~~ (Outdated)
  - Probabilistic Robotics [The MIT Press, 2005] (Outdated, but still the best bible)
  - Past, Present, and Future of SLAM [T-RO, 2016]
  - The Future of Real-time SLAM @ ICCV 2015 (mostly focused on visual SLAM)
- Codes
  - Github: https://github.com/topics/slam
    - Visual SLAM: ORB-SLAM2 (mono, stereo, RGB-D), DSO, VINS-Mono (mono+IMU), RTAB-Map (RGB-D), ...
    - LiDAR SLAM: GMapping, Cartographer, ...
    - Optimizer (backend): g2o, GTSAM, Ceres Solver, ...
  - ~~OpenSLAM~~ (outdated), MRPT
  - Base libraries: OpenCV, PCL (Point Cloud Library), Open3D
  - My open tutorial on 3D vision for beginners (contains basics for visual odometry and SLAM)
- Communities
  - SLAM KR (Korean Facebook group)

SLAM with a Gaussian Filter



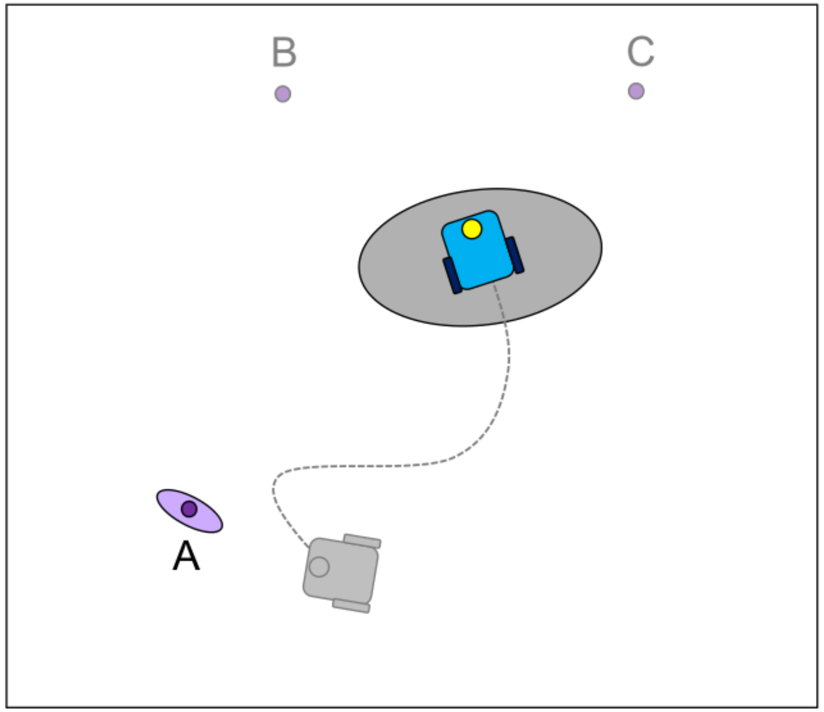
Start: robot has zero uncertainty



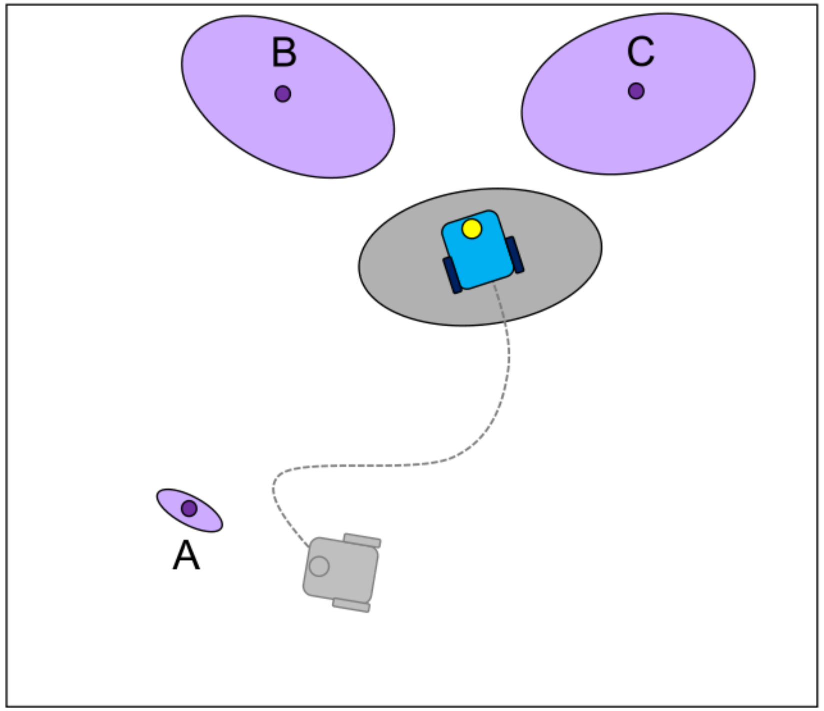
First measurement of feature A

Image: ETHZ's Autonomous Mobile Robots lectures by Prof. Chli

SLAM with a Gaussian Filter



Robot moves forwards: uncertainty grows

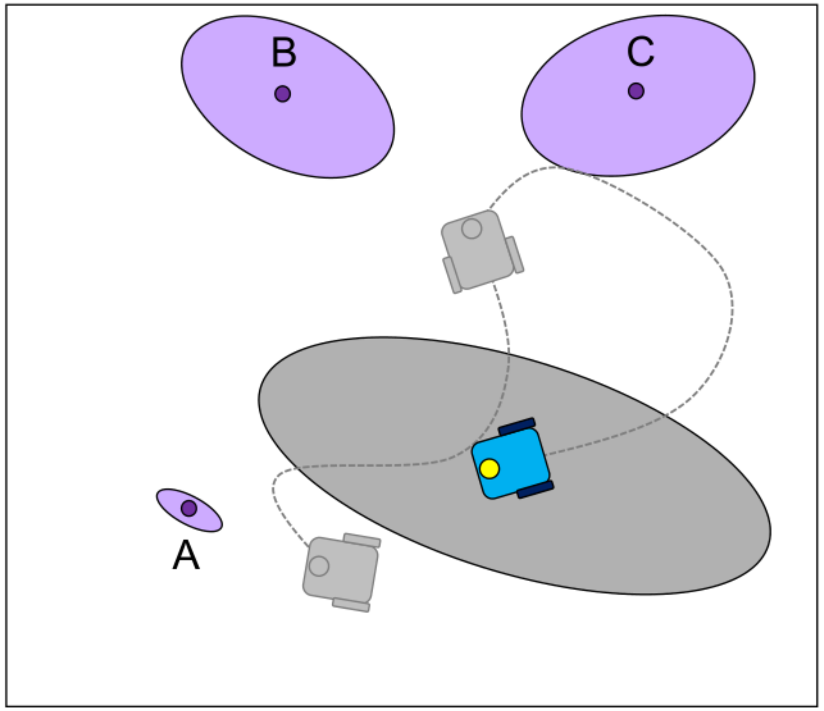


Robot makes first measurements of B & C

Image: ETHZ's Autonomous Mobile Robots lectures by Prof. Chli

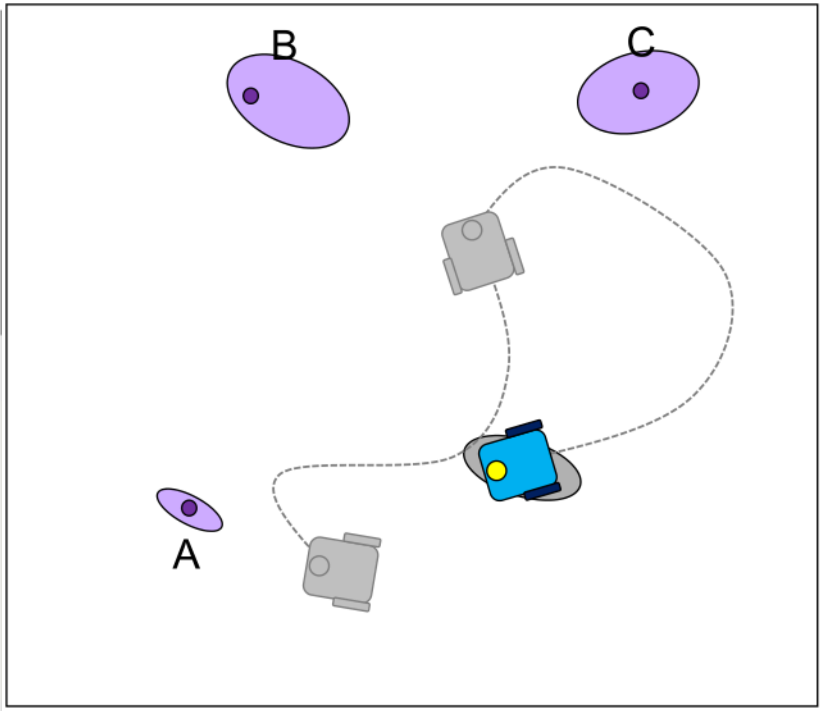
SLAM with a Gaussian Filter

Predict how the robot has moved



Robot moves again: uncertainty grows more

Correct the robot pose and map

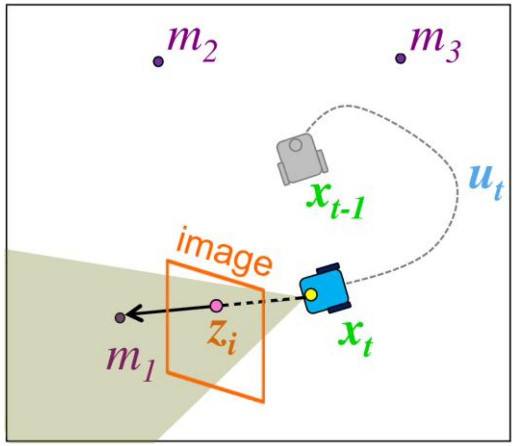


Robot re-measures A: “**loop closure**”
uncertainty shrinks

Image: ETHZ‘s Autonomous Mobile Robots lectures by Prof. Chli

SLAM in Graphical Representation

- **Notation**
  - $\boldsymbol{x}_t$: Robot pose at time t / $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_t\}$: Robot path
  - m_i: i-th feature / $\{m_0, m_1, \cdots, m_N\}$: Map
  - $\boldsymbol{u}_t$: Robot motion between $t-1$ and t (a.k.a. control input)
  - $\boldsymbol{z}_i$: Observation of i-th feature



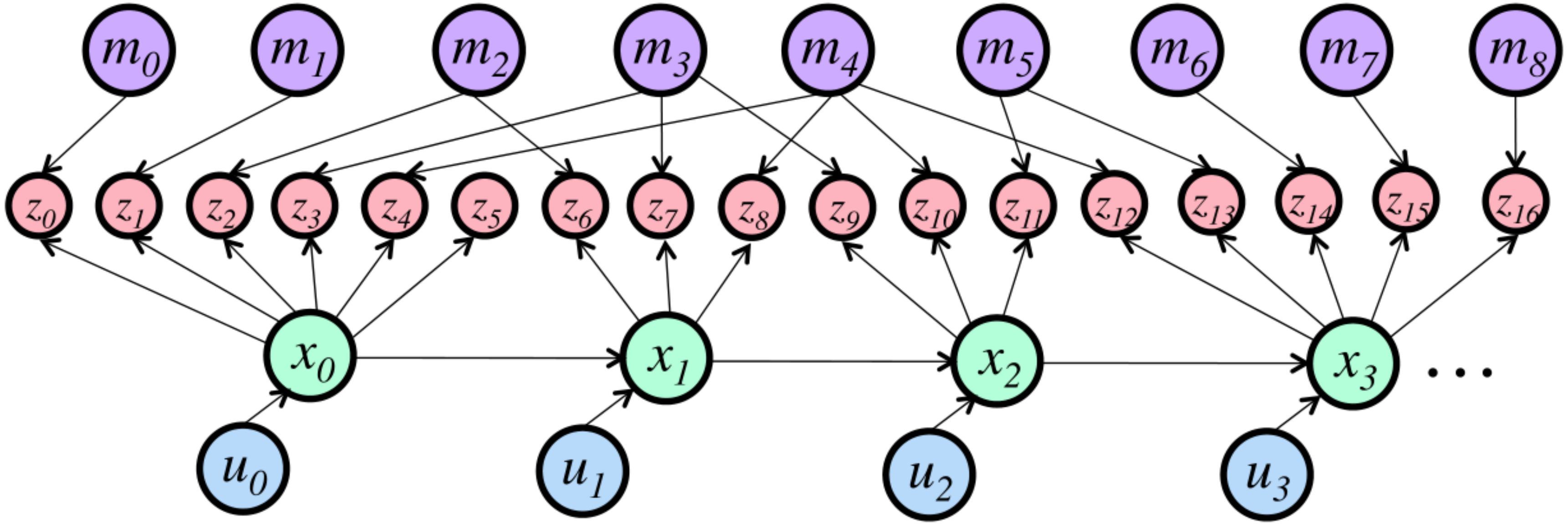


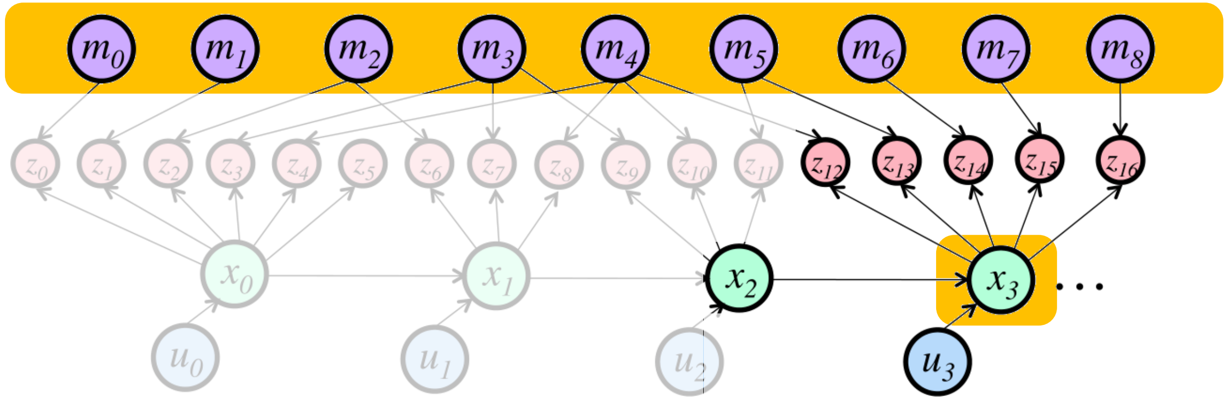
Image: ETHZ's Autonomous Mobile Robots lectures by Prof. Chli

Problem Formulation

- **Online SLAM** estimates most recent pose and map

  ~ Maximize the posterior $P(\boldsymbol{x}_t, m_{0:N} | \boldsymbol{z}_{0:k}, \boldsymbol{u}_{1:t})$ or more simply $P(\boldsymbol{x}_t, m_{n:N} | \boldsymbol{z}_{n:k}, \boldsymbol{u}_t, \boldsymbol{x}_{t-1})$

Markov assumption



- **Full SLAM** estimates entire path and map

  ~ Maximize the posterior $P(\boldsymbol{x}_{0:t}, m_{0:N} | \boldsymbol{z}_{0:k}, \boldsymbol{u}_{1:t})$

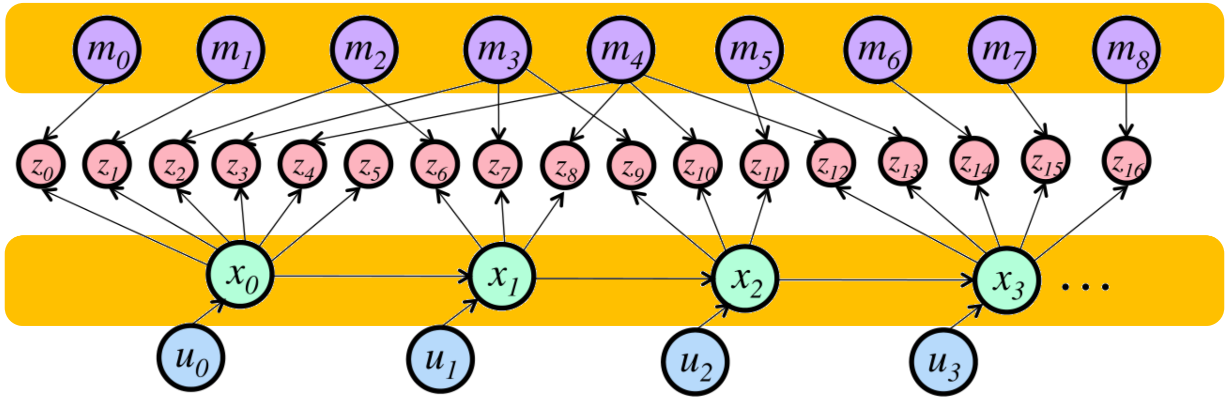
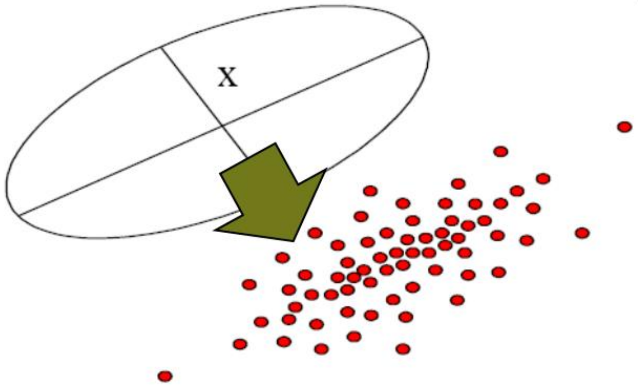


Image: ETHZ's Autonomous Mobile Robots lectures by Prof. Chli

Bayesian Filtering



- Approaches
  - Follow **prediction** (with motion) and **correction** (with observation) steps
  - Use probabilistic representation
    - Kalman filter: Gaussian / Particle filter: a set of samples
  - Usually based on Markov assumption
- Pros
  + **Run online** (but it does not mean real-time)
- Cons
  - **Does not scale to high-dimensional problems**
  - Kalman filter: Unimodal / Particle filter: Need many particles for good convergence

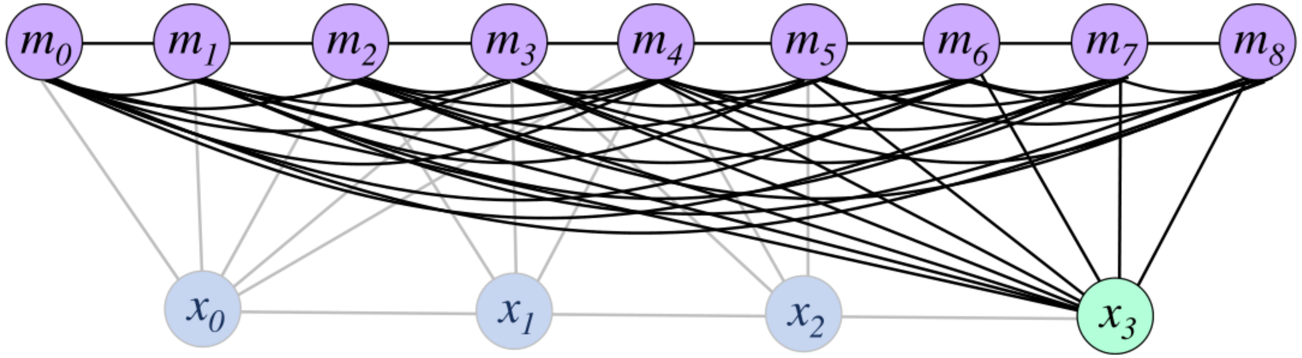


Image: ETHZ's Autonomous Mobile Robots lectures by Prof. Chli

Graph Optimization

- Approaches
  - Minimize the nonlinear least-squares cost function (~ reprojection error)
  - Use a batch maximum likelihood (ML) approach
  - Assume Gaussian noise distribution
- Pros
  + **Information can move backward**
  + Best possible results given from the data and models
- Cons
  - **Computational burden**
  - Difficult to provide the online result for control

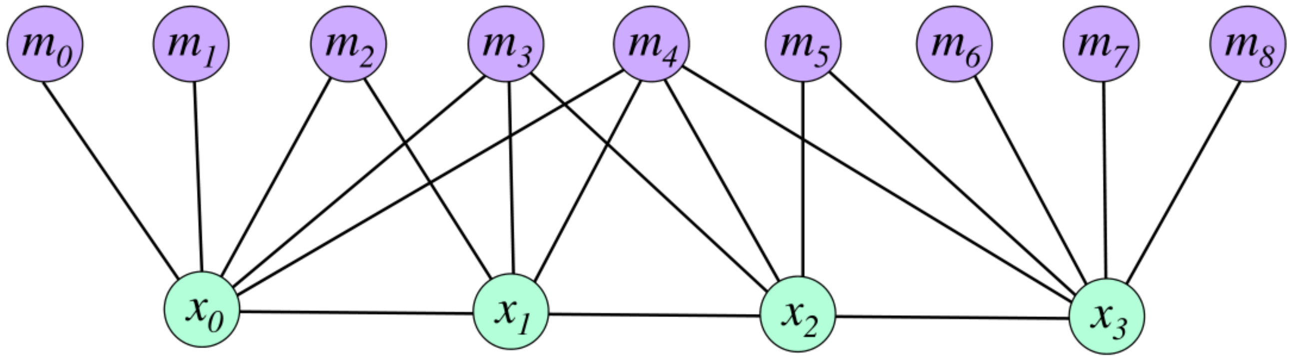
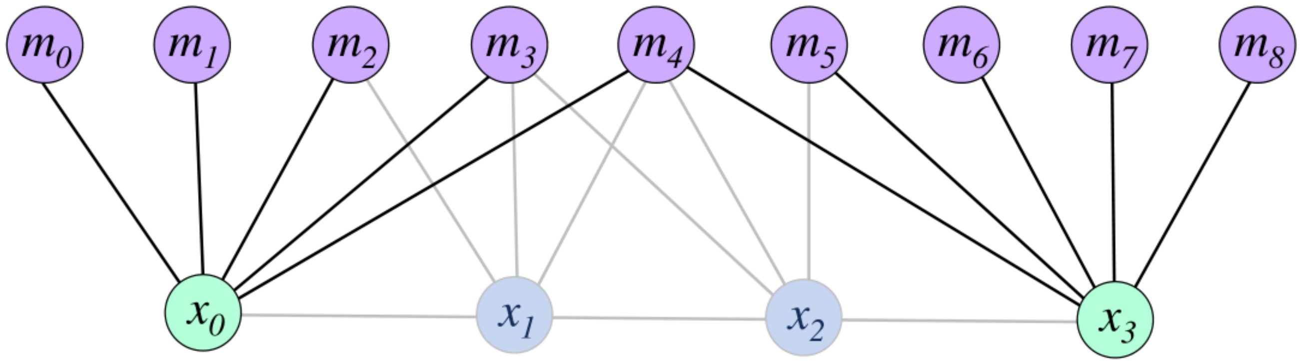


Image: ETHZ's Autonomous Mobile Robots lectures by Prof. Chli

Graph Optimization

- Approaches
  - Minimize the nonlinear least-squares cost function (~ reprojection error)
  - Use a batch maximum likelihood (ML) approach
  - Assume Gaussian noise distribution
- Pros
  + **Information can move backward**
  + Best possible results given from the data and models
- Cons
  - **Computational burden** → Sparsify the graph or apply sliding window or parallelize the burden
  - Difficult to provide the online result for control



Keyframe-based SLAM

Image: ETHZ's Autonomous Mobile Robots lectures by Prof. Chli

Graph Optimization

- Approaches
  - Minimize the nonlinear least-squares cost function (~ reprojection error)
  - Use a batch maximum likelihood (ML) approach
  - Assume Gaussian noise distribution
- Pros
  - \+ **Information can move backward**
  - \+ Best possible results given from the data and models
- Cons
  - **Computational burden** → Sparsify the graph or apply sliding window or parallelize the burden
  - Difficult to provide the online result for control

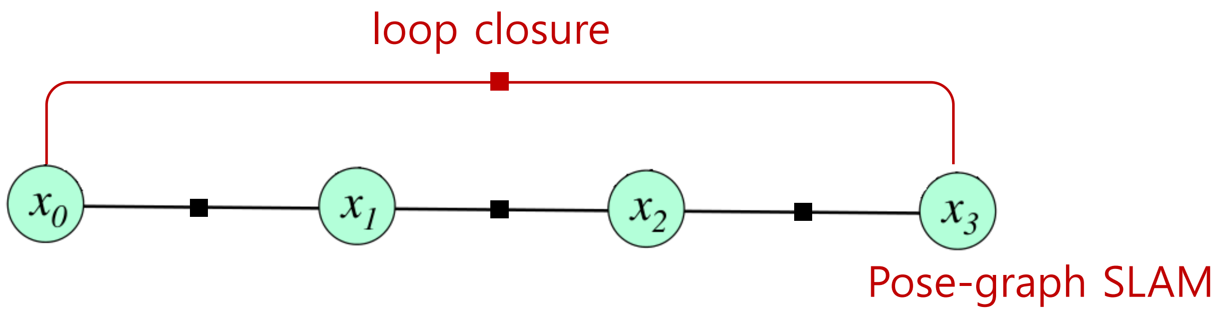
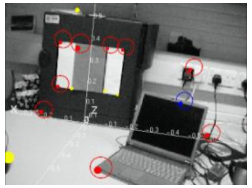


Image: ETHZ's Autonomous Mobile Robots lectures by Prof. Chli

Visual Odometry and SLAM: History

Monocular Camera

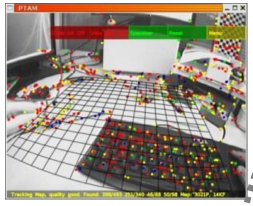
SLAM in Robotics



MonoSLAM (2003): features, Bayesian filtering

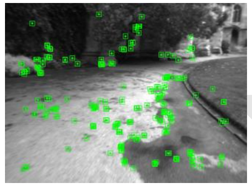
v.s.

SfM in Computer Vision



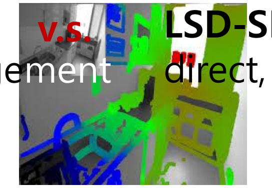
PTAM (2007): more features, bundle adjustment (keyframes)

DTAM (2011): dense volumetric



ORB-SLAM (2014)
better feature management

v.s.



LSD-SLAM (2014)
direct, semi-dense

RGB-D Camera

KineticFusion (2011): dense volumetric

ElasticFusion (2015)
dense surfels

DynamicFusion (2015)
dynamic dense

Paradigm #1: Bayesian Filtering v.s. Bundle Adjustment

Robotics

Localization Mapping

SLAM

(Simultaneous Localization and Mapping)

Bayesian Filtering

e.g. **MonoSLAM**

[Davison et al., T-PAMI, 2007]

VS

Computer Vision

Motion Estimation Reconstruction

SfM

(Structure from Motion)

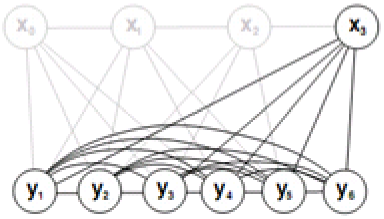
Bundle Adjustment

e.g. **PTAM**

[Klein and Murray, ISMAR, 2007]

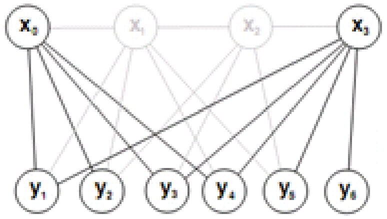
"Real-time Monocular SLAM: Why Filter?"

[Strasdat et al., ICRA, 2010]



≤

1. Global optimization
2. # of features (100 vs 4000)



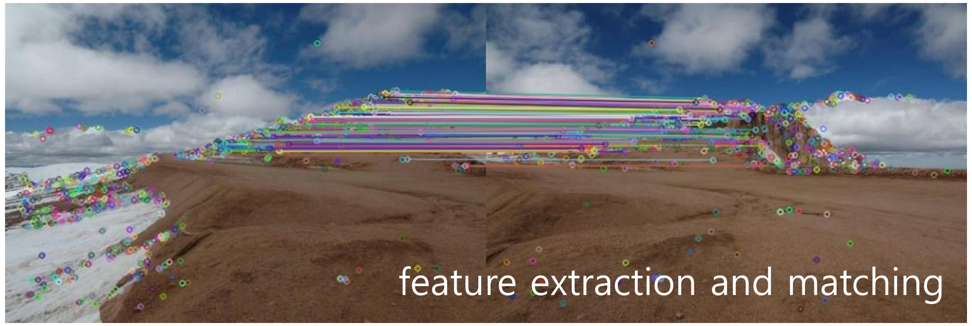
Paradigm #2: Feature-based Method v.s. Direct Method

- e.g. Image stitching

+

=

Feature-based Method **VS** **Direct Method**



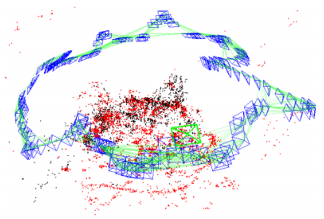
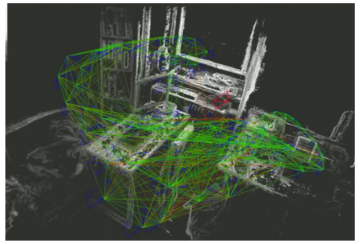
$$\arg\min_{\mathrm{H}} \sum_i \left\| \mathrm{H}\mathbf{x}_i - \mathbf{x}'_i \right\|^2$$

$$\arg\min_{\mathrm{H}} \sum_{u,v} \left\| I\left(\left[\begin{smallmatrix}u\\v\\1\end{smallmatrix}\right]\right) - I'\left(\mathrm{H}\left[\begin{smallmatrix}u\\v\\1\end{smallmatrix}\right]\right) \right\|^2$$

Paradigm #2: Feature-based Method v.s. Direct Method

Feature-based/Direct SLAM

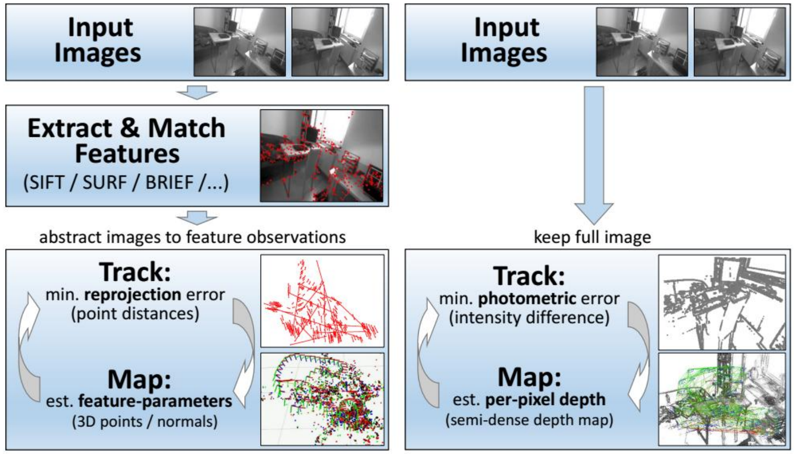
Universidad Zaragoza

Feature-Based SLAM	Direct SLAM
	
Minimize **Feature Reprojection Error**	Minimize **Photometric Error**
Sparse Reconstruction	**Semi Dense / Dense** Reconstruction
PTAM, ORB-SLAM	**DTAM, LSD-SLAM, DPPTAM**

ROBOTICS UNIVERSIDAD DE ZARAGOZA

Comparison

Feature-Based | **Direct**



Chiuso '02, Nistér '04, Eade '06, Klein '06, Davison '07, Strasdat '10, Mur-Artal '14, ...

Matthies '88, Hanna '91, Comport '06, Newcombe '11, Engel '13, ...

Why should we still use features?

Universidad Zaragoza

Robustness

- **Reliable two-view monocular initialization**
- **Good invariance to viewpoint and illumination**
- **Less affected by auto-gain and auto-exposure**
- **Less affected by dynamic elements**

Accuracy

- **Bundle adjustment (joint map-trajectory optimization)**

Place Recogniton (loop detection, relocalization)

- **Bags of Words**

But sparse reconstructions ...

ROBOTICS UNIVERSIDAD DE ZARAGOZA

Comparison

Feature-Based	Direct
can only use & reconstruct corners	can use & reconstruct whole image
faster	slower (but good for parallelism)
flexible: outliers can be removed retroactively.	inflexible: difficult to remove outliers retroactively.
robust to inconsistencies in the model/system (rolling shutter).	not robust to inconsistencies in the model/system (rolling shutter).
decisions (KP detection) based on less complete information.	decision (linearization point) based on more complete information.
no need for good initialization.	needs good initialization.
~20+ years of intensive research	~4 years of research (+5years 25 years ago)

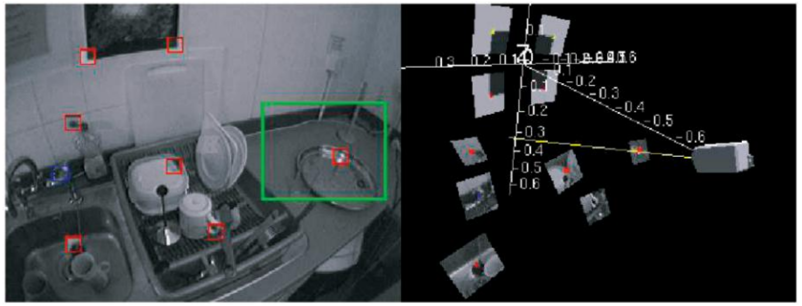
MonoSLAM (2003)

- **The first successful visual SLAM with pure vision, drift-free, and real-time ability**

- **References**
  - Code repositories
    - SceneLib v1, Andrew Davison (LGPL)
    - SceneLib v2, Hanme Kim (MIT license)
    - MonoSLAM Implementation in ROS, rrg-polito (MPL)
  - Papers
    - Andrew Davison, *Real-Time Simultaneous Localisation and Mapping with a Single Camera*, ICCV, 2003 DOI PDF
    - Andrew Davison et al., *MonoSLAM: Real-Time Single Camera SLAM*, T-PAMI, 2007 DOI PDF

EKF-SLAM

- State variable (robot state, feature map): $\mathbf{x} = [\mathbf{x}_v, \mathbf{y}_1, \mathbf{y}_2, \ldots]^\top$
  - Robot state (position, quaternion, linear velocity, angular velocity): $\mathbf{x}_v = [\mathbf{r}^W, \mathbf{q}^{WR}, \mathbf{v}^W, w^R]^\top$
  - Feature map (position, direction): $\mathbf{y}_i = \left[\mathbf{r}_i^W, \mathbf{h}_i^W\right]$
    - $\mathbf{h}_i^W$: A unit vector describing the feature direction (~ normal vector)



- Motion model: **Constant velocity motion model**

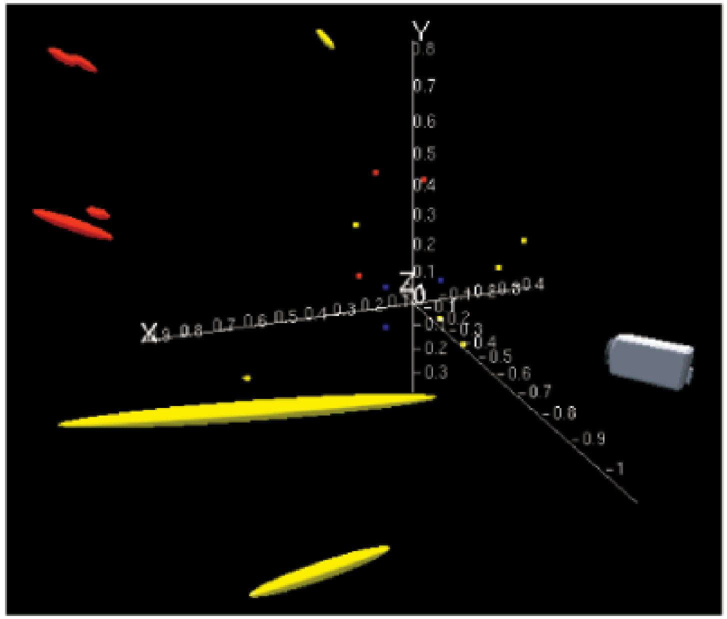
$$\mathbf{f}_v = \begin{bmatrix} \mathbf{r}_{new}^W \\ \mathbf{q}_{new}^{WR} \\ \mathbf{v}_{new}^W \\ w_{new}^R \end{bmatrix} = \begin{bmatrix} \mathbf{r}^W + (\mathbf{v}^W + \mathbf{V}^W)\Delta t \\ \mathbf{q}^{WR} \times \mathbf{q}((w^R + \Omega^R)\Delta t) \\ \mathbf{v}^W + \mathbf{V}^W \\ w^R + \Omega^R \end{bmatrix} \text{ and } \mathrm{Q}_v = \frac{\partial \mathbf{f}_v}{\partial \mathbf{n}} \mathrm{P}_n \frac{\partial \mathbf{f}_v}{\partial \mathbf{n}}^\top \text{ where noise } \mathbf{n} = \begin{bmatrix} \mathbf{V}^W \\ \Omega^R \end{bmatrix} \text{ and its variance } \mathrm{P}_n$$

- Observation model: **Pinhole camera model** with **radial distortion**

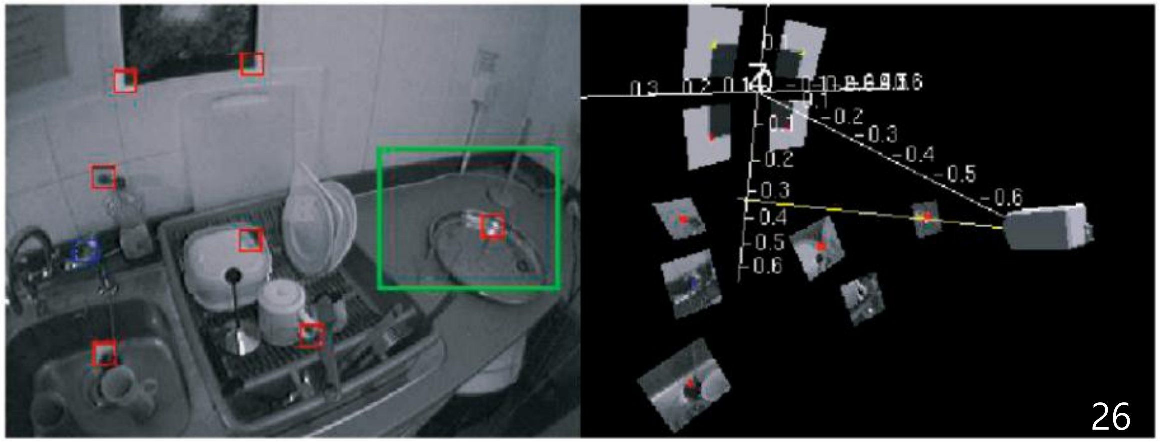
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_0 + f_u \frac{h_{Lx}^R}{h_{Lz}^R} \\ v_0 + f_v \frac{h_{Ly}^R}{h_{Lz}^R} \end{bmatrix} \text{ with } \mathbf{u}_{di} = \begin{bmatrix} u_d \\ v_d \end{bmatrix} = \begin{bmatrix} u_0 + \frac{u - u_0}{\sqrt{1 + 2K_1 r^2}} \\ v_0 + \frac{v - v_0}{\sqrt{1 + 2K_1 r^2}} \end{bmatrix} \text{ where } \mathbf{h}_L^R = \mathrm{R}^{RW}(\mathbf{r}_i^W - \mathbf{r}^W) \text{ and } r^2 = (u - u_0)^2 + (v - v_0)^2$$

$$\mathrm{S}_i = \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{x}_v} \mathrm{P}_{xx} \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{x}_v}^\top + \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{x}_v} \mathrm{P}_{xr_i} \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{r}_i}^\top + \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{r}_i} \mathrm{P}_{r_i x} \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{x}_v}^\top + \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{r}_i} \mathrm{P}_{r_i r_i} \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{r}_i}^\top + \mathrm{R}$$

Feature and Map



- Feature detection
  - Feature point: [Good-feature-to-track]
  - Feature descriptor: 11x11-pixel patch
    - MonoSLAM does *not* update the saved templates for features over time.
  - [Bucketing] (for feature addition): 80x60-pixel box (~ 4x4 boxes)
- Feature matching
  - Feature prediction (by camera projection)
  - Feature uncertainty: 2-by-2 covariance matrix S_i (in pixel domain)
  - *Active* elliptical search: Window size 3σ
    - MonoSLAM removes a feature from the map when feature is failed with less than 50%.
- Feature orientation estimation
- Map initialization
  - Starting from the known object (typically 4 features)



Experiments

- Configuration
  - Camera: 320x240 with 30 Hz
    - $f_u = f_v = 195$ pixels, $(u_0, v_0) = (162, 125)$, and $K_1 = 6 \times 10^{-6}$
    - Nearly 100 degrees FOV
  - Map size: <u>100 features</u>
  - Processing time: Approx. 52.6 Hz
    - Observation size: <u>10-12 features per a frame</u>

Image loading and administration	2 ms
Image correlation searches	3 ms
Kalman Filter update	5 ms
Feature initialization search	4 ms
Graphical rendering	5 ms
Total	**19 ms**

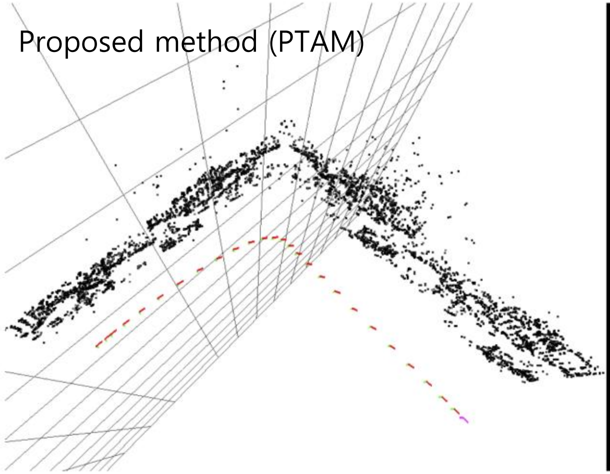
PTAM (2007)

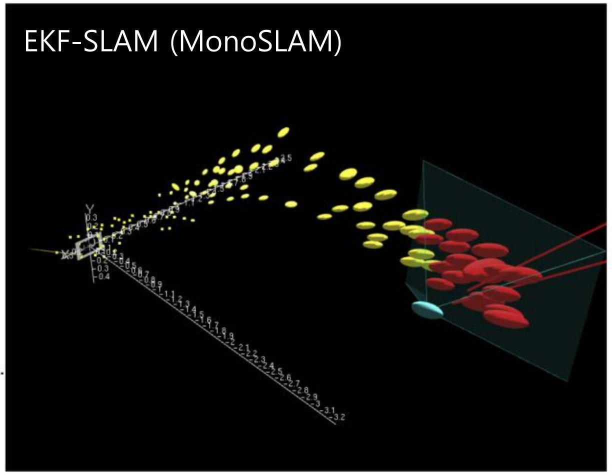
- **The <u>first</u> successful <u>keyframe-based BA</u> approach with <u>real-time ability</u>**

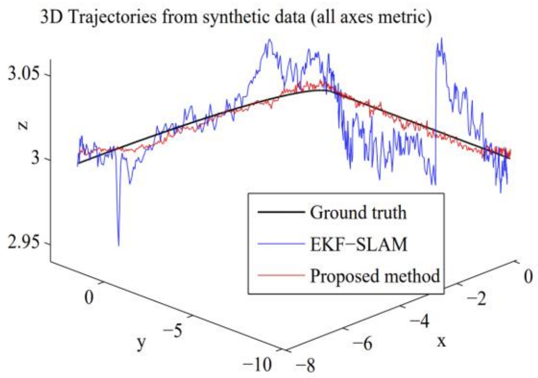
- **References**
  - Code repositories
    - PTAM-GPL, Oxford-PTAM (GPL v3)
    - The modified version of PTAM for ROS, ETHZ-ASL
  - Papers
    - Georg Klein and David Murray, *Parallel Tracking and Mapping for Small AR Workspaces*, ISMAR, 2007 DOI PDF Slides

PTAM (2007)

- **PTAM (Parallel Tracking and Mapping)**
  - Feature: FAST-10 corner (8x8-pixel patch, SSD matching)
  - Tracking
    - Coarse-to-fine tracking
    - Computing time: 20 ms with 4000 features
  - Mapping
    - Keyframe-based bundle adjustment with 5-point algorithm + RANSAC
    - Computing time: 1.7 sec with 50-99 keyframes







ORB-SLAM Series (2015, 2017, 2021)

- **The best successor of PTAM with support of large-scale spaces and long-term operation**
- **References**
  - Code repositories
    - ORB-SLAM (monocular), Raul Mur-Artal (GPL v3)
    - ORB-SLAM2 (monocular, stereo, and RGB-D), Raul Mur-Artal (GPL v3)
      - ROS wrappers: appliedAI-Initiative, ethz-asl
    - ORB-SLAM3 (visual, visual-inertial, and multi-map), UZ-SLAMLab (GPL v3)
    - Project webpage, Raul Mur-Atral
  - Papers
    - Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos, *ORB-SLAM: A Versatile and Accurate Monocular SLAM System*, T-RO, 2015 DOI arXiv
    - Raul Mur-Artal and Juan D. Tardos, *ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras*, T-RO, 2017 DOI arXiv
    - Carlos Campos et al., *ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM*, T-RO, 2021 DOI arXiv

ORB-SLAM (2015)

- **Overview**
  - Feature: ORB **(Oriented FAST and Rotated BRIEF) for all tasks**
    - Note) SIFT/SURF (~300 ms), A-KAZE (~100 ms), ORB (~33 ms), BRIEF/LDB (rotation variant)
  - **Three threads**: Tracking, local mapping, and loop closing

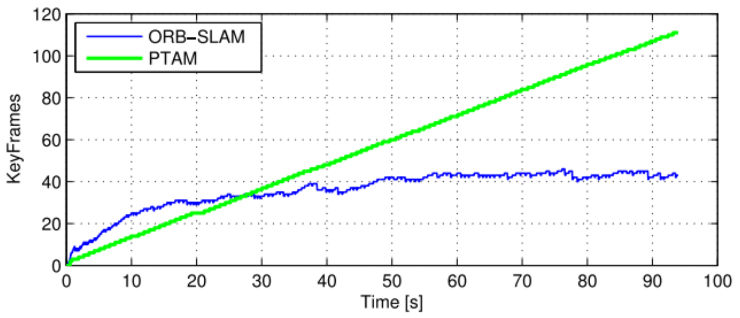
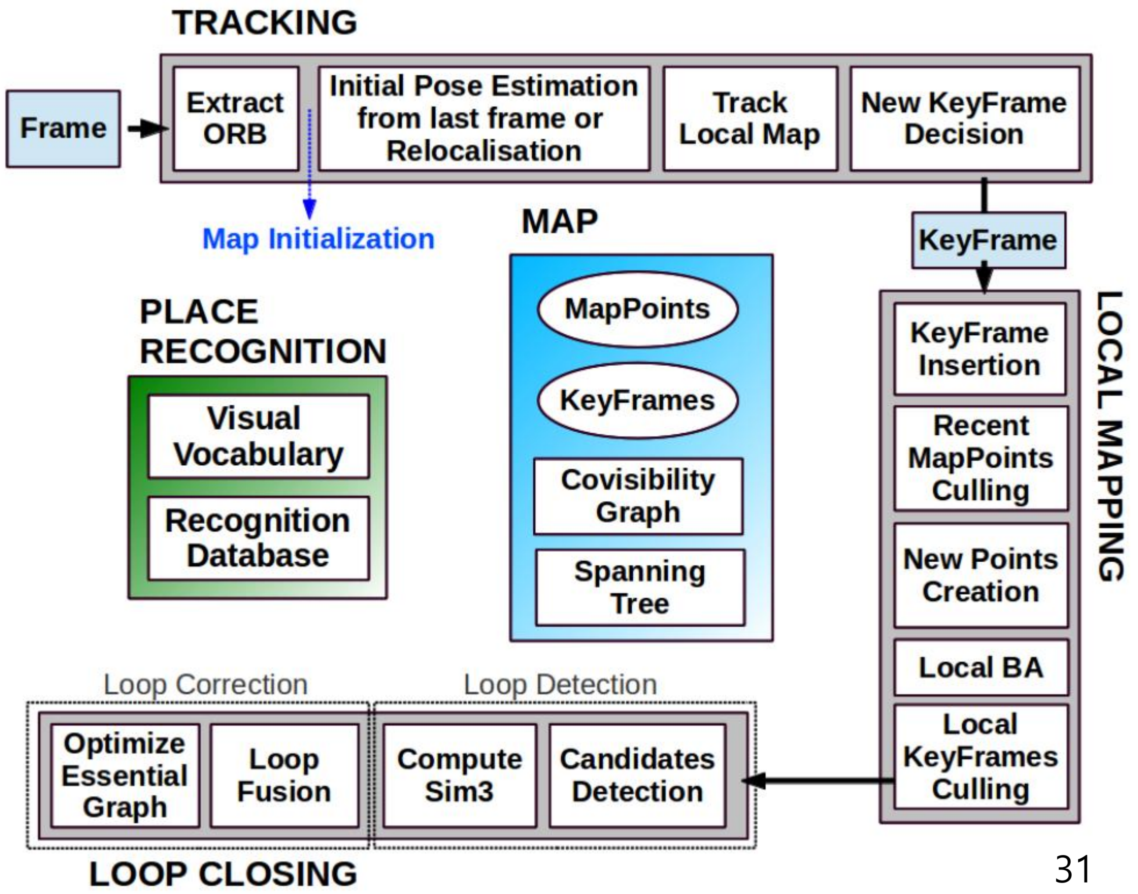
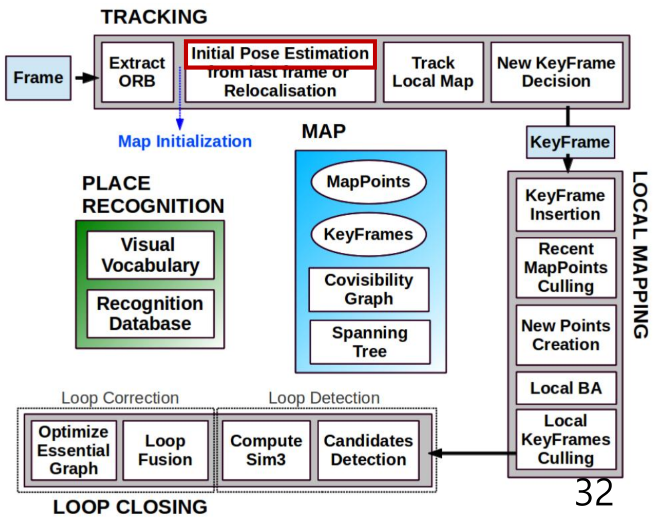


Fig. 9. Lifelong experiment in a static environment where the camera is always looking at the same place from different viewpoints. PTAM is always inserting keyframes, while ORB-SLAM is able to prune redundant keyframes and maintains a bounded-size map.



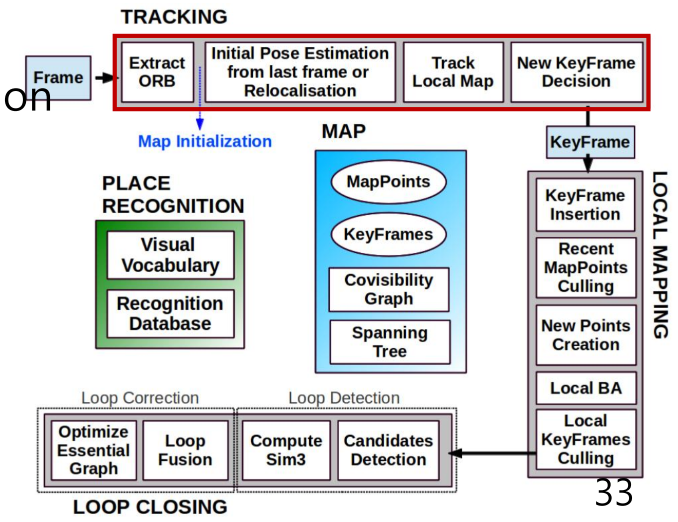
ORB-SLAM (2015)

- **Map initialization**
  - Two models: 1) homography for planar scenes and 2) fundamental matrix for general scenes
  - Model parameter estimation: RANSAC (MSAC)
    - Symmetric transfer errors S
    - Threshold: Chi-square test at 95% (H = 5.99, F = 3.84 @ σ = 1 pixel)
  - Model selection: $\frac{S_H}{S_H+S_F} > 0.45$ → (nearly) planar and low parallax → homography
    - Map initialization is delayed until enough parallax and low reprojection error.
  - Map refinement: *Full* BA



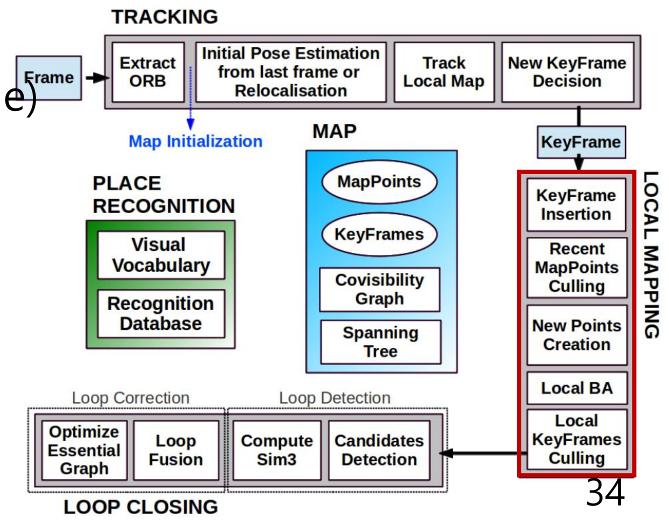
ORB-SLAM (2015)

- **Tracking**
  - ORB extraction: [FAST] corners (scale levels: 8, scale factor: 1.2) with [ORB] descriptors
    - 2000 corners for 1241x376 resolution, 1000 corners 512x384 to 752x480 resolutions
    - Bucketing for uniform feature distribution (max 5 features/cell)
  - Initial pose estimation
    - Tracking: Constant velocity motion model → *motion-only* BA using the previous frame
    - Global relocalization (if lost): RANSAC and EPnP algorithm for each keyframe
  - Pose refinement: *Motion-only* BA using a local visible map
  - New keyframe selection if
    - Passing more than 20 frames from last global relocalization
    - Local mapping is idle or passing more than 20 frames from last keyframe insertion
    - Tracking at least 50 points
  - Tracking less than 90% points than the reference keyframe.



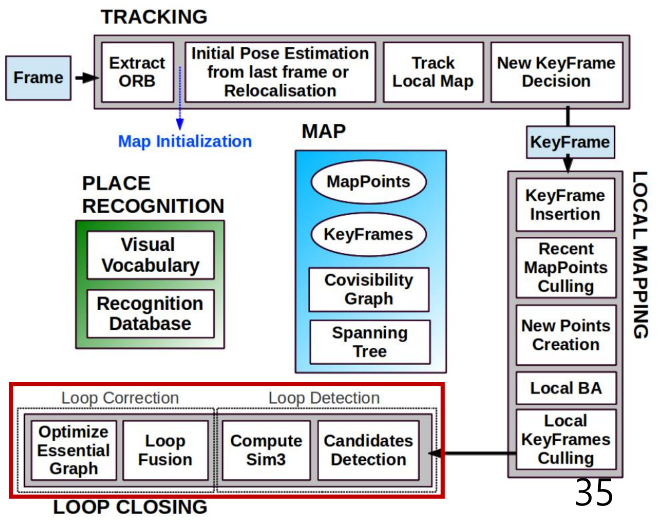
ORB-SLAM (2015)

- **Local mapping**
  - Keyframe insertion: Updating covisibility graph, spanning tree, and bag-of-words
  - Recent map point culling if not
    - Tacking and visible more than 25%
    - Observable at least 3 keyframes
  - New map point addition: Triangulation if
    - Positive depth
    - Satisfying parallax, reprojection error, and scale consistency condition
  - *Local* BA (for the current keyframe and its connecting keyframes)
  - Local keyframe culling if
    - 90% of map points are visible in at least 3 other keyframes (in same or finer scale)



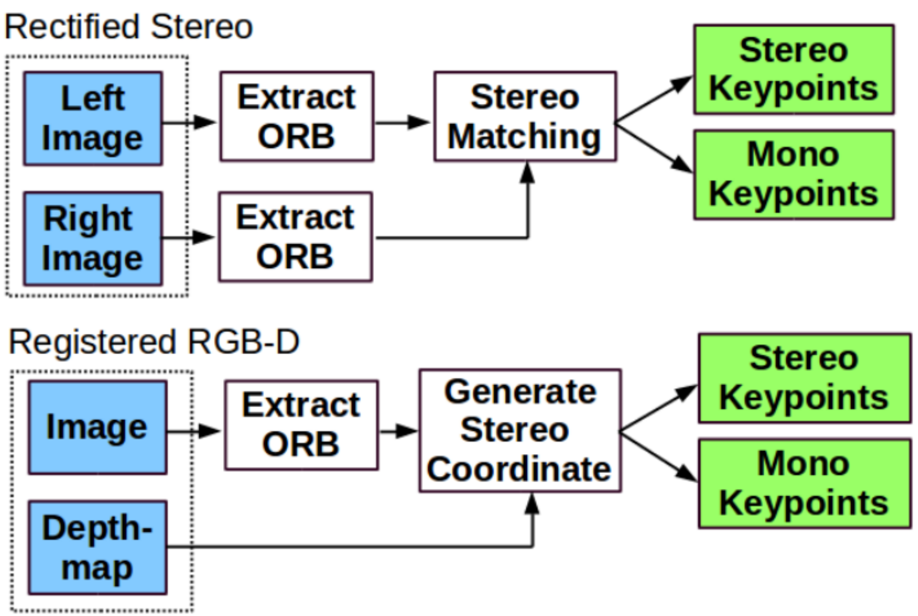
ORB-SLAM (2015)

- **Loop closing (of the last keyframe)**
  - Loop candidate detection: DBoW2
    - The candidates should be consistent (~ at least 3 consecutively connected in the covisibility graph).
  - Similarity transformation computation: RANSAC with 3D points → optimization over inliers
    - Why similarity? (Why 7 DOF?) 3 for translation, 3 for rotation, and 1 for scale (due to scale drift)
  - Loop fusion: Fusing duplicated map points and adding new edges in the covisibility graph
  - Essential graph optimization: Pose graph optimization over similarity transformation, Sim(3)



ORB-SLAM2 (2017)

- **An ORB-SLAM extension for <u>stereo and RGB-D cameras</u>**
- **Two types of keypoints for different**
  - Camera configurations: Monocular, stereo, and RGB-D cameras
  - **Monocular keypoints**: $\mathbf{x}_m = (u_L, v_L)$
    - Projection function: $\pi_m(\mathbf{X}) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix}$ where $\mathbf{X} = (X, Y, Z)$
  - **Stereo keypoints**: $\mathbf{x}_s = (u_L, v_L, u_R)$
    - Projection function: $\pi_s(\mathbf{X}) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ f_x \frac{X-b}{Z} + c_x \end{bmatrix}$
    - Far stereo keypoints are considered as monocular keypoints whose criteria is 40 times of baseline, b.
- After loop closing, ORB-SLAM2 performs **pose graph optimization over rigid-body transform** instead of similarity transform (due to no drift error).



ORB-SLAM3 (2021)

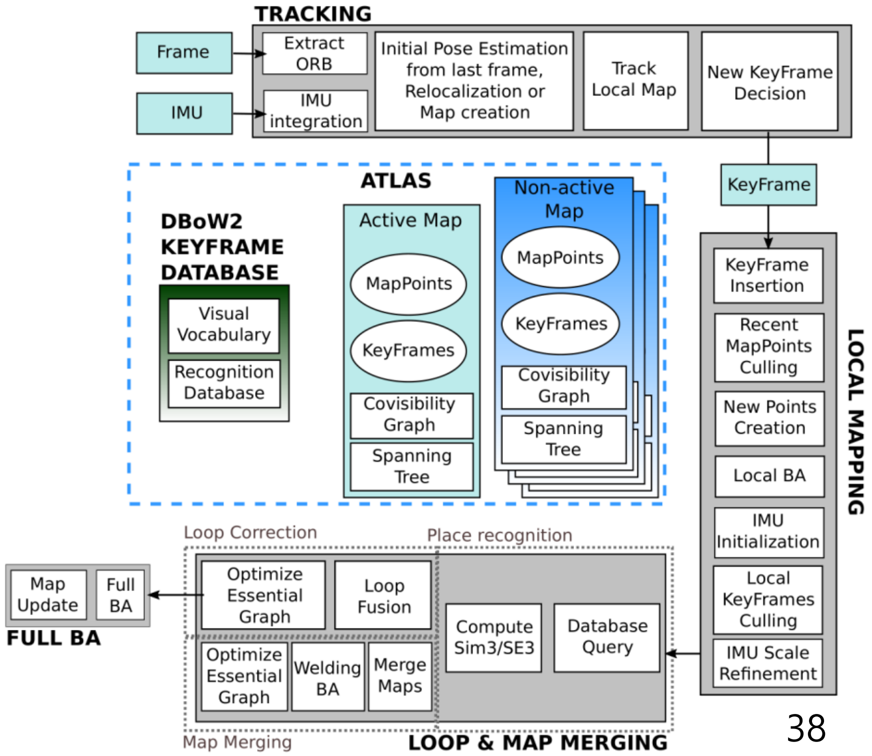
- **An ORB-SLAM2 extension for 1) fisheye cameras and 2) inertia sensors with 3) multi-session support**

	SLAM or VO	Pixels used	Data association	Estimation	Relocalization	Loop closing	Multi Maps	Mono	Stereo	Mono IMU	Stereo IMU	Fisheye	Accuracy	Robustness	Open source
Mono-SLAM [13], [14]	SLAM	Shi Tomasi	Correlation	EKF	-	-	-	✓	-	-	-	-	Fair	Fair	[15][1]
PTAM [16]–[18]	SLAM	FAST	Pyramid SSD	BA	Thumbnail	-	-	✓	-	-	-	-	Very Good	Fair	[19]
LSD-SLAM [20], [21]	SLAM	Edgelets	Direct	PG	-	FABMAP PG	-	✓	✓	-	-	-	Good	Fair	[22]
SVO [23], [24]	VO	FAST+ Hi.grad.	Direct	Local BA	-	-	-	✓	✓	-	-	✓	Very Good	Very Good	[25][2]
ORB-SLAM2 [2], [3]	SLAM	ORB	Descriptor	Local BA	DBoW2	DBoW2 PG+BA	-	✓	✓	-	-	-	Exc.	Very Good	[26]
DSO [27]–[29]	VO	High grad.	Direct	Local BA	-	-	-	✓	✓	-	-	✓	Fair	Very Good	[30]
DSM [31]	SLAM	High grad.	Direct	Local BA	-	-	-	✓	-	-	-	-	Very Good	Very Good	[32]
MSCKF [33]–[36]	VO	Shi Tomasi	Cross correlation	EKF	-	-	-	✓	-	✓	✓	-	Fair	Very Good	[37][3]
OKVIS [38], [39]	VO	BRISK	Descriptor	Local BA	-	-	-	-	-	✓	✓	✓	Good	Very Good	[40]
ROVIO [41], [42]	VO	Shi Tomasi	Direct	EKF	-	-	-	-	-	✓	✓	✓	Good	Very Good	[43]
ORBSLAM-VI [4]	SLAM	ORB	Descriptor	Local BA	DBoW2	DBoW2 PG+BA	-	✓	-	✓	-	-	Very Good	Very Good	-
VINS-Fusion [7], [44]	VO	Shi Tomasi	KLT	Local BA	DBoW2	DBoW2 PG	✓	-	✓	✓	✓	✓	Good	Exc.	[45]
VI-DSO [46]	VO	High grad.	Direct	Local BA	-	-	-	-	-	✓	-	-	Very Good	Exc.	-
BASALT [47]	VO	FAST	KLT (LSSD)	Local BA	-	ORB BA	-	-	-	-	✓	✓	Very Good	Exc.	[48]
Kimera [8]	VO	Shi Tomasi	KLT	Local BA	-	DBoW2 PG	-	-	-	-	✓	-	Good	Exc.	[49]
ORB-SLAM3 (ours)	SLAM	ORB	Descriptor	Local BA	DBoW2	DBoW2 PG+BA	✓	✓	✓	✓	✓	✓	Exc.	Exc.	[5]

[1] Last source code provided by a different author. Original software is available at [50].
[2] Source code available only for the first version, SVO 2.0 is not open source.
[3] MSCKF is patented [51], only a re-implementation by a different author is available as open source.

ORB-SLAM3 (2021)

- **Camera models**
  - A pinhole camera model and Kannala-Brandt fisheye model
  - Relocalization: EPnP → MLPnP (using projective rays as input)
  - Non-rectified stereo
- **Visual-Inertia optimization**
  - State vector: $S_i = \{\mathbf{R}_i, \mathbf{p}_i, \mathbf{v}_i, \mathbf{b}_i^g, \mathbf{b}_i^a\}$ (orientation, position, velocity, gyroscope bias, and accelerometer bias)
  - Reprojection error: $\mathbf{r}_{ij} = \mathbf{x}_{ij} - \pi\left(\mathbf{T}_{CB}\mathbf{T}_i^{-1} \oplus \mathbf{X}_j\right)$ where $\mathbf{T}_i = [\mathbf{R}_i, \mathbf{p}_i] \ \in \mathrm{SE}(3)$
  - Inertia residual (Δ: preintegrated term)
    - Orientation residual: $\mathbf{r}_{\Delta\mathbf{R}_{i,i+1}} = \mathrm{Log}\left(\Delta\mathbf{R}_{i,i+1}^{\mathsf{T}}\mathbf{R}_i^{\mathsf{T}}\mathbf{R}_{i+1}\right)$
    - Position residual: $\mathbf{r}_{\Delta\mathbf{p}_{i,i+1}} = \mathbf{R}_i^{\mathsf{T}}(\mathbf{p}_{i+1} - \mathbf{p}_i - \mathbf{v}_i\Delta t - \frac{1}{2}\mathbf{g}\Delta t^2) - \Delta\mathbf{p}_{i,i+1}$
    - Velocity residual: $\mathbf{r}_{\Delta\mathbf{v}_{i,i+1}} = \mathbf{R}_i^{\mathsf{T}}(\mathbf{v}_{i+1} - \mathbf{v}_i - \mathbf{g}\Delta t) - \Delta\mathbf{v}_{i,i+1}$
- **Multi-session operation**
  - …



Experiments

- **ORB-SLAM**
  - TUM RGB-D dataset
    - ORB-SLAM was more accurate than PTAM and LSD-SLAM.
  - KITTI odometry dataset
    - ORB-SLAM was failed in KITTI 01 (highway sequence).
- **ORB-SLAM2**
  - KITTI odometry dataset
    - ORB-SLAM2 (stereo) was more accurate than stereo LSD-SLAM.
    - ORB-SLAM2 (stereo) overcame KITTI 01 (highway sequence).
  - TUM RGB-D dataset
    - ORB-SLAM (RGB-D) was more accurate than ElasticFusion, Kintinuous, DVO-SLAM, and RGB-D SLAM.
- **ORB-SLAM3**
  - ...

Why Visual Odometry?

Visual odometry

vs

Wheel odometry

+ direct motion measure
+ six degree-of-freedoms
+ easy to install
– heavy computation
– visual disturbance (e.g. moving objects)

– indirect motion measure (e.g. slippage)
– two degree-of-freedoms
– necessary to be on rotor/shaft
+ simple calculation

Why Visual Odometry?

Visual odometry

vs

Wheel odometry

– indirect motion measure (e.g. slippage)
– two degree-of-freedoms
– necessary to be on rotor/shaft
+ simple calculation

no wheels / rough terrains, in-the-sky, under-the-water

Why Visual Odometry?

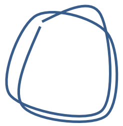
Visual odometry **VS**

Wheel odometry

– indirect motion measure (e.g. slippage)
– two degree-of-freedoms
– necessary to be on rotor/shaft
\+ simple calculation

no wheels / rough terrains, in-the-sky, under-the-water

Visual Odometry **VS**



Visual SLAM

no assumption on trajectories
→ navigation / large space (outdoor)

closed-loop is preferred for convergence
→ mapping / small space (indoor, campus)

Why Visual Odometry?

Visual odometry

VS

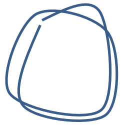
Wheel odometry

– indirect motion measure (e.g. slippage)
– two degree-of-freedoms
– necessary to be on rotor/shaft
+ simple calculation

no wheels / rough terrains, in-the-sky, under-the-water

Visual Odometry

VS



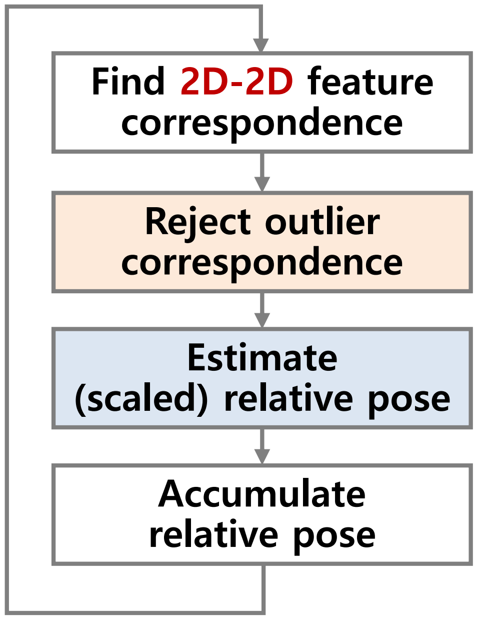
Visual SLAM

closed-loop is preferred for convergence
→ mapping / small space (indoor, campus)

real navigation situations

Feature-based Monocular Visual Odometry

Two-view Motion Estimation



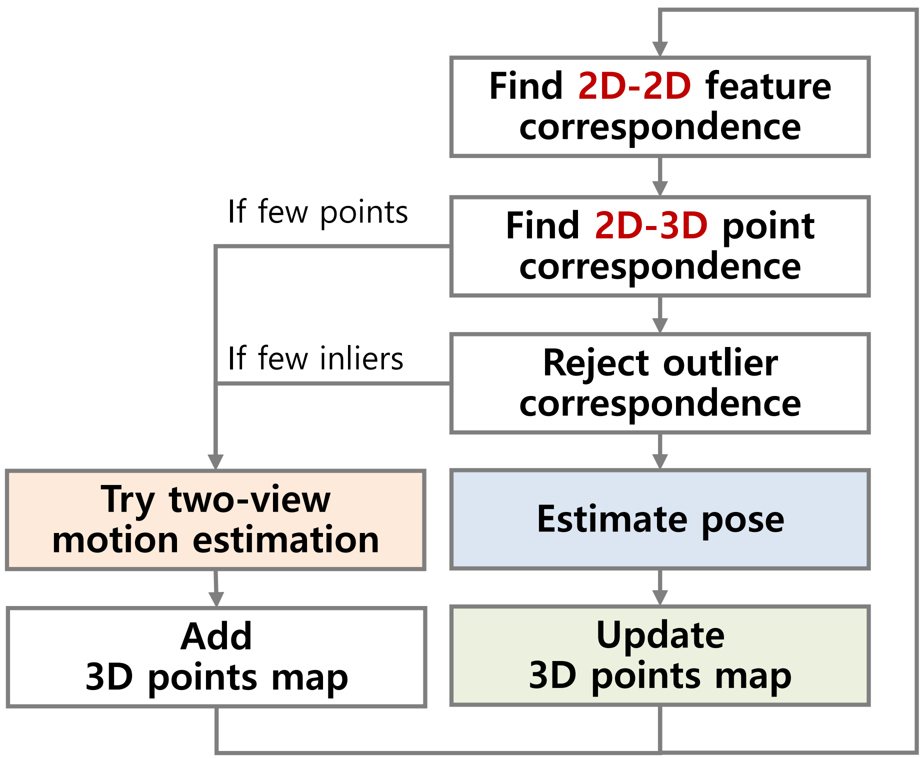
- Feature: **Good-Feature-to-Track** [Shi94_CVPR]
  with bucketing to distribute features
- Correspondence: **Lucas-Kanade optical flow** [Lucas81_IJCAI]

- Adaptive MSAC [Choi09_IROS]
- **Iterative 5-point algorithm** [Choi15_IJCAS]
- Error measure: Sampson distance

- Normalized 8-point algorithm
- Scale-from-ground with **asymmetric kernels** [Choi13_URAI]

Feature-based Monocular Visual Odometry

PnP Pose Estimation



- Feature: **Good-Feature-to-Track** [Shi94_CVPR] with bucketing to distribute features
- Correspondence: **Lucas-Kanade optical flow** [Lucas81_IJCAI]

- Adaptive MSAC
- **Iterative PnP algorithm** (3-point algorithm)
- Error measure: Projection error

- Iterative PnP algorithm
- Scale-from-ground with **asymmetric kernels** [Choi13_URAI]

- **Bundle adjustment** over last K keyframes
  Reprojection error with Cauchy loss function

Feature-based Monocular Visual Odometry