ORBSLAM

Robotic Localization and Mapping

16833

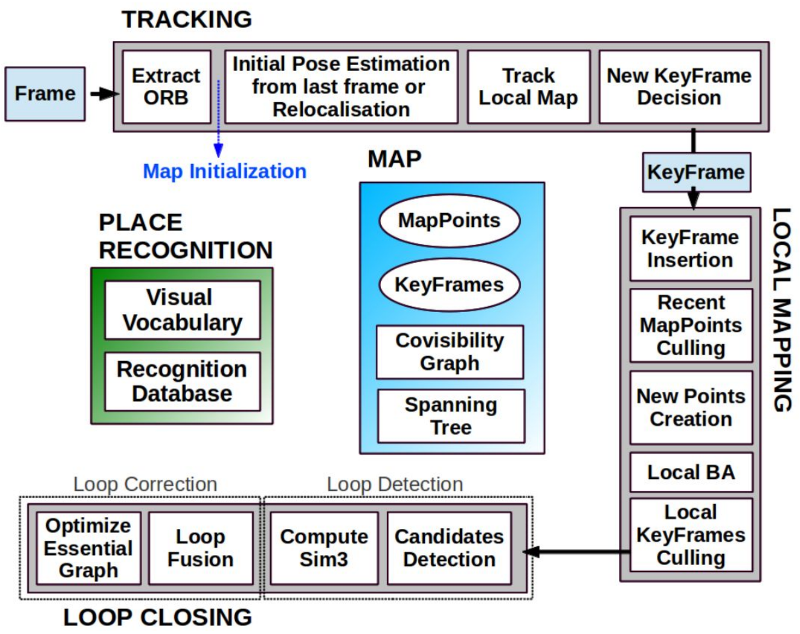
Fall 2024

Dan McGann

Slides adapted Sudharshan Suresh and Paloma Sodhi

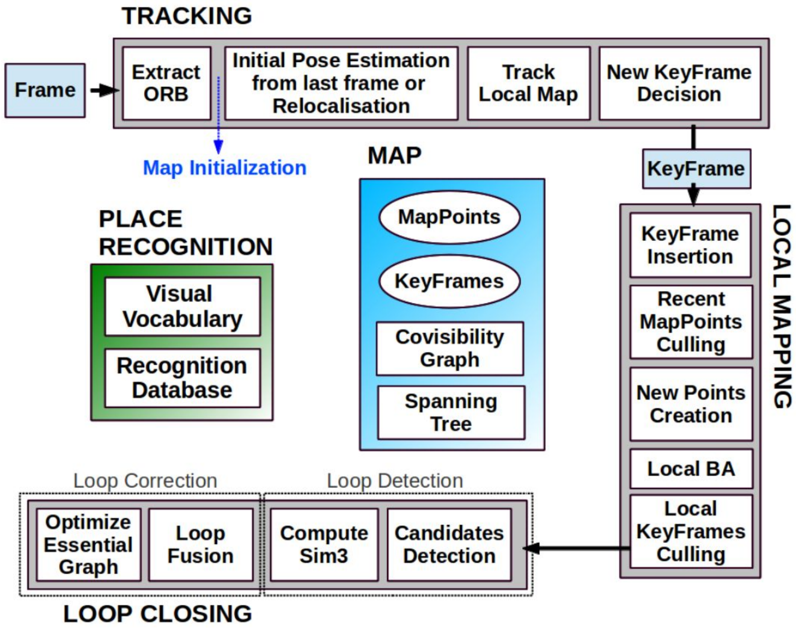
Outline:

1. Motivation
2. Definitions
3. Tracking
4. Local Mapping
5. Loop Closing



Outline:

1. **Motivation**
2. Definitions
3. Tracking
4. Local Mapping
5. Loop Closing



ORBSLAM

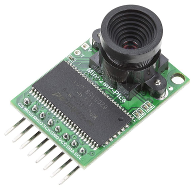
- Homework 3: SLAM Solvers
  - Landmarks are 2d points, in the real world these will often be 3d
  - Odometry is given as relative poses, in the real world we are not given this

- How do we actually compute Odometry? What can we actually use as landmarks? How do we associate landmarks? How can we do all of this in real-time?

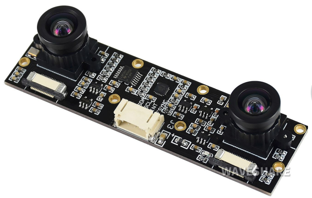
Enter ORBSLAM

Options for Sensors

- Odometry must be derived from raw data gathered by sensors on your robot



Monocular Camera

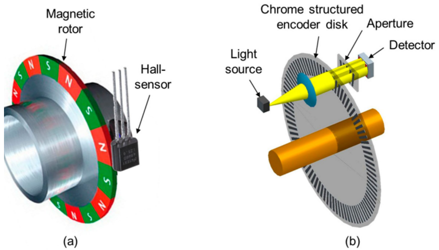


Stereo Camera

Depth Camera

LIDAR

GPS

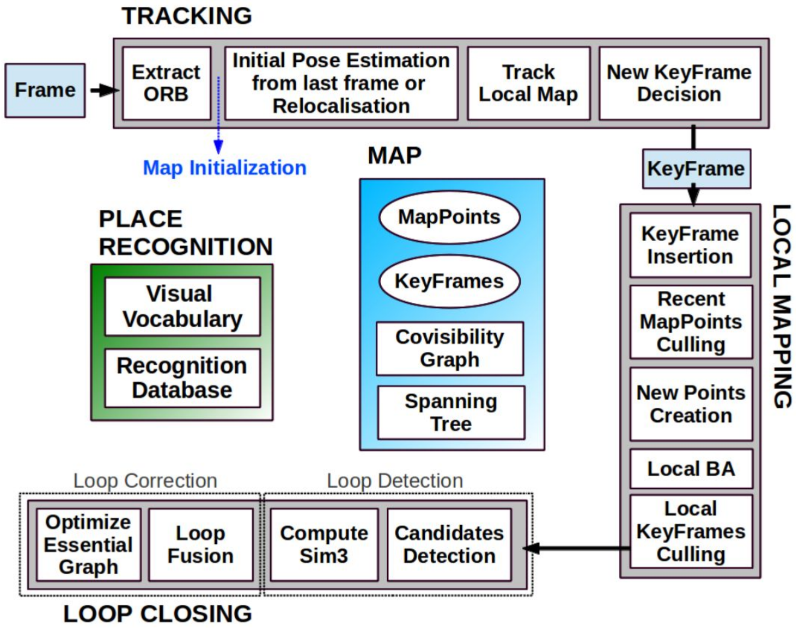


Encoders

- Choice of sensors depends on (weight, size, cost, operation environment, desired accuracy, efficiency of available algorithms, etc).

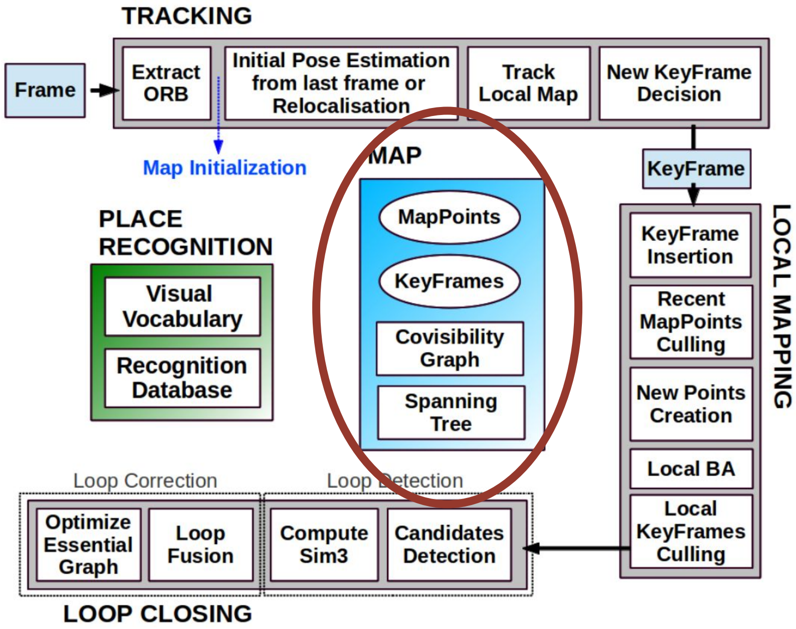
Outline:

1. **Motivation**
2. Definitions
3. Tracking
4. Local Mapping
5. Loop Closing



Outline:

1. Motivation
2. **Definitions**
3. Tracking
4. Local Mapping
5. Loop Closing



ORB-SLAM Definitions / Goal

- Def Map: A set of feature points $M = \{p\}$
  - Where each point p consists of
    - A 3d point in the map frame [x, y, z]
    - An average viewing direction
    - An average ORB descriptor

- Def KeyFrames: A set of camera poses $T \in SIM(3)$
  - Where each frame has an associated image
    - Implicit: A set of associated feature points
    - Implicit: Camera intrinsics

Simultaneously Localize (compute *T*) and Map (compute *M*)

ORB-SLAM Sub-Definitions

- “Co-Visible”: Two Keyframes are co-visible if they are associated to a common feature point in the map

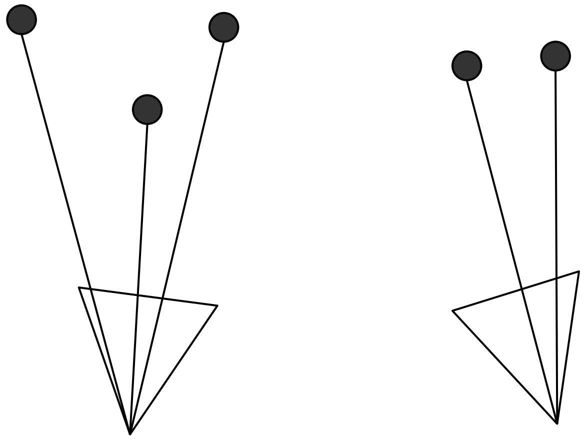
- “Co-Visibility Graph”: A graph $G \in \{V, E\}$ where
  - Vertices = Keyframes
  - An edge exists between two vertices iff the two keyframes are co-visible

- “Essential Graph”: A graph $G \in \{V, E\}$ that
  - Minimal-ish spanning graph of the co-visibility graph containing
    - The strongest co-visibility edges
    - Loop Closure edges

Co-Visibility Simple Example

Feature Points

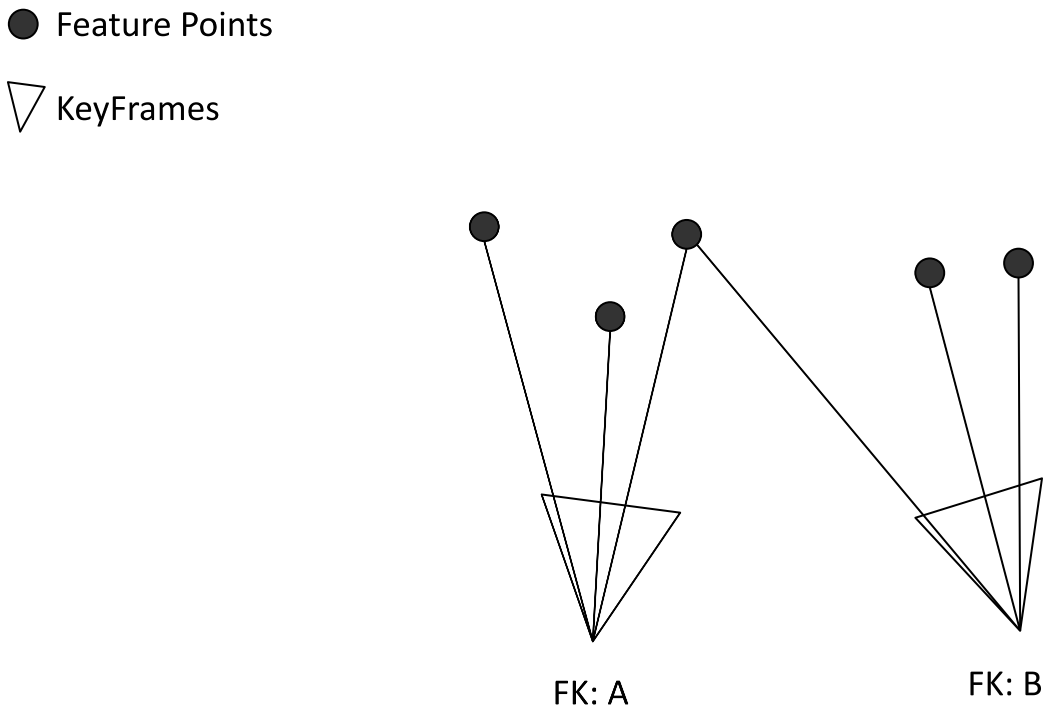
KeyFrames



FK: A FK: B

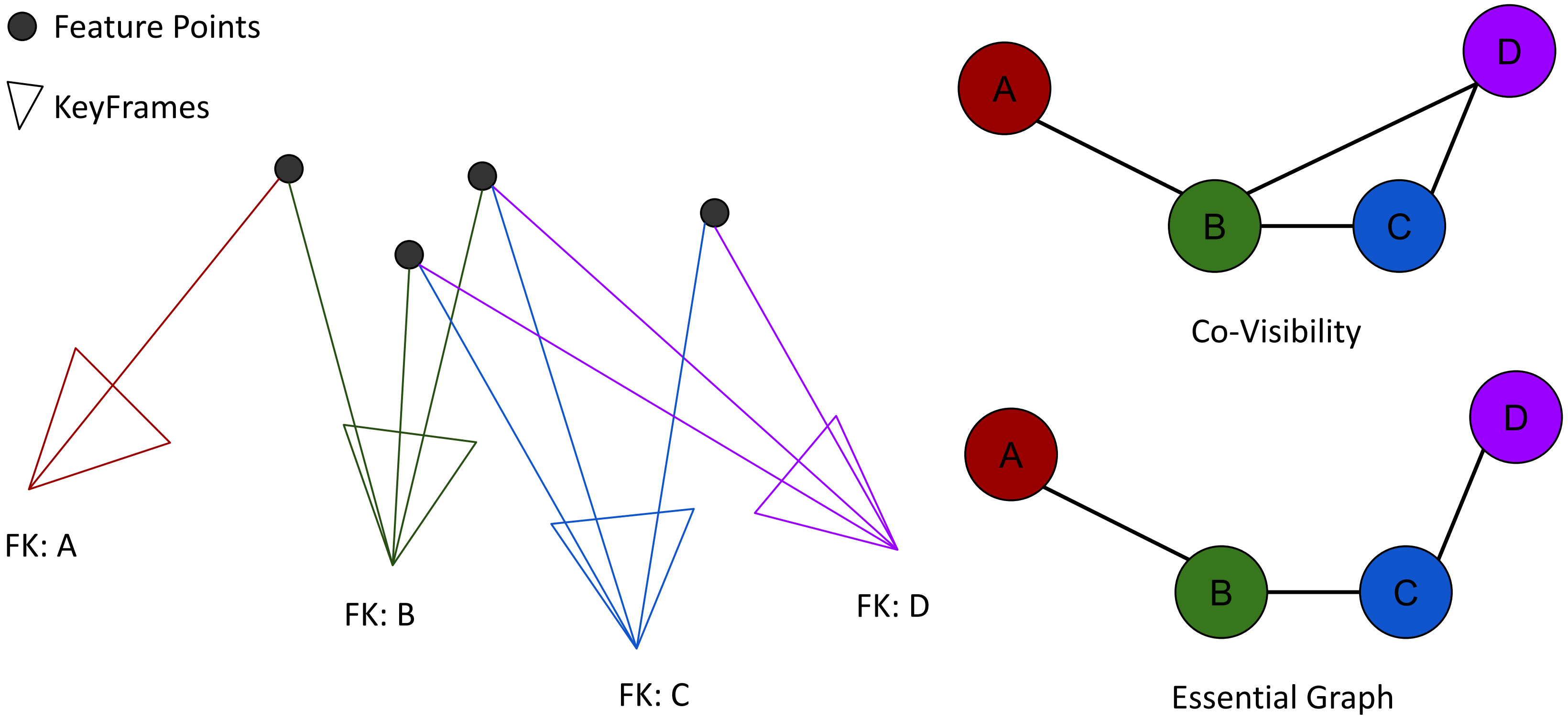
A/B are NOT Co-Visible

Co-Visibility Simple Example



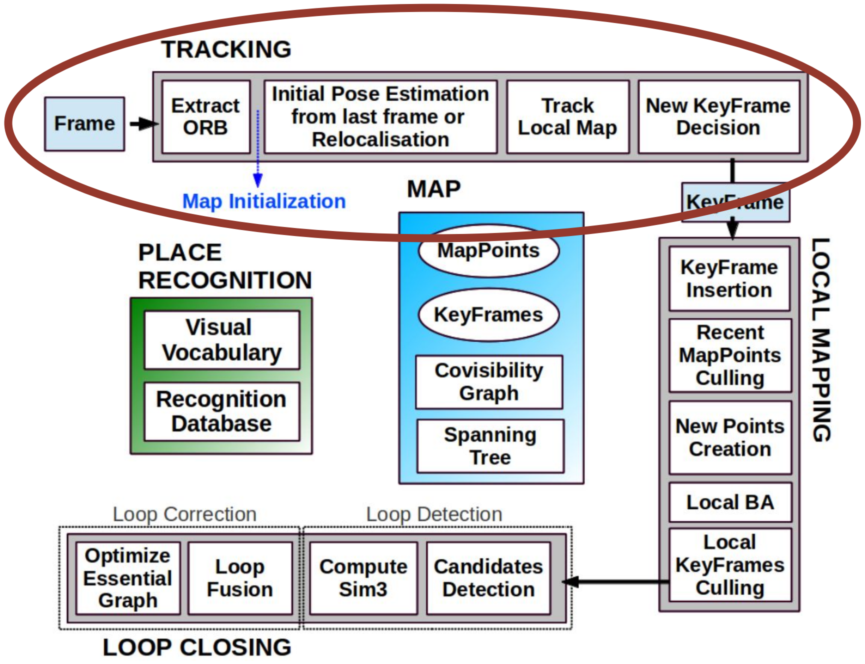
A/B ARE
Co-Visible

Co-Visibility / Essential Graph Example

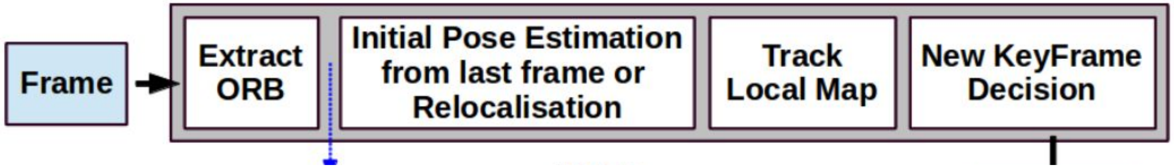


Outline:

1. Motivation
2. Definitions
3. **Tracking**
4. Local Mapping
5. Loop Closing

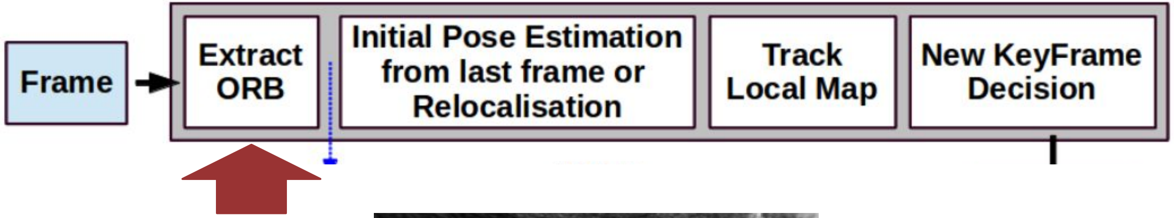


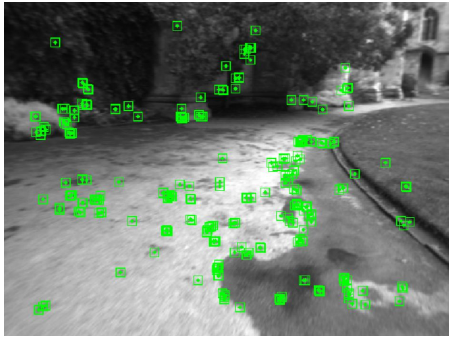
Local Tracking: Goal



- Given a new frame from the camera estimate the camera's position.
  - Computing odometry between two frames
  - Implicitly: Computing odometry between keyframes
- Occurs at high rate (image capture rate)

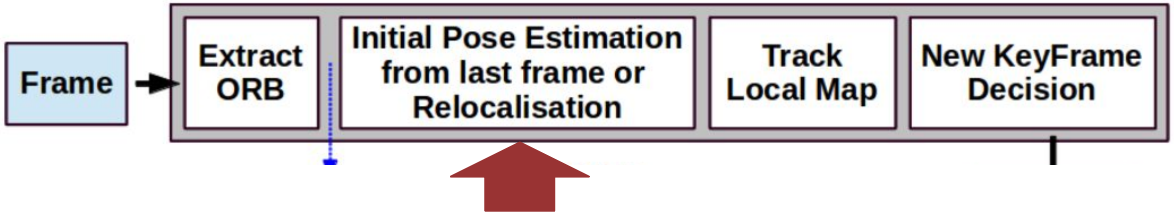
Local Tracking: Feature Extraction





- ORB Features
  - Selected at time because fastest to compute + rotationally invariant
  - Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary R. Bradski: "*ORB: An efficient alternative to SIFT or SURF*". ICCV 2011: 2564-2571.

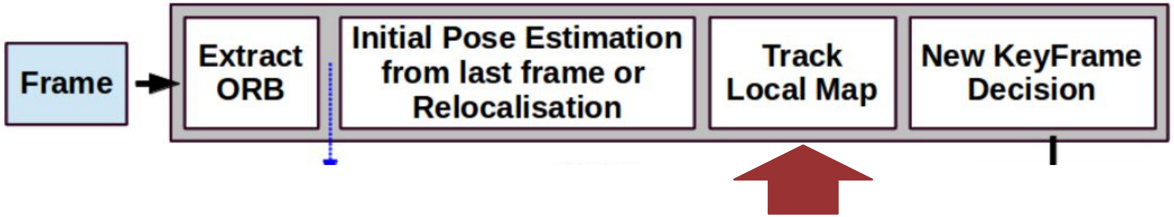
Local Tracking: Feature Extraction



- Case 1: Tracking - Assume constant velocity and integrate to get new position
  - Condition: Last frame tracked properly

- Case 2: Relocalization - [Covered Later]
  - Condition: Last frame tracking failed

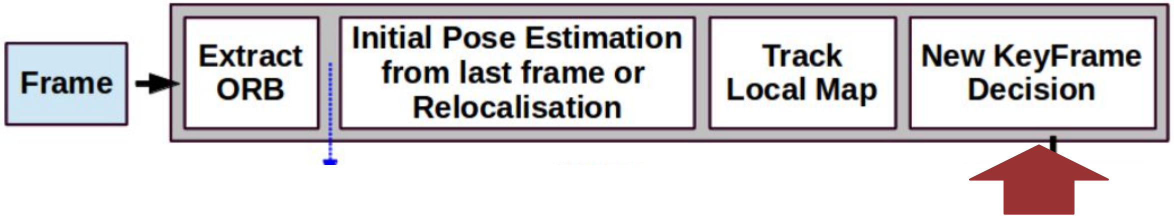
Local Tracking: Feature Extraction



1. With initial pose, we can associate frame-features with map-points
2. We can solve for new position with ***motion-only BA***
    a. Optimizes the following cost (nonlinear)

$$\mathbf{e}_{i,j} = \mathbf{x}_{i,j} - \pi_i(\mathbf{T}_{iw}, \mathbf{X}_{w,j})$$

Local Tracking: KeyFrame Decision

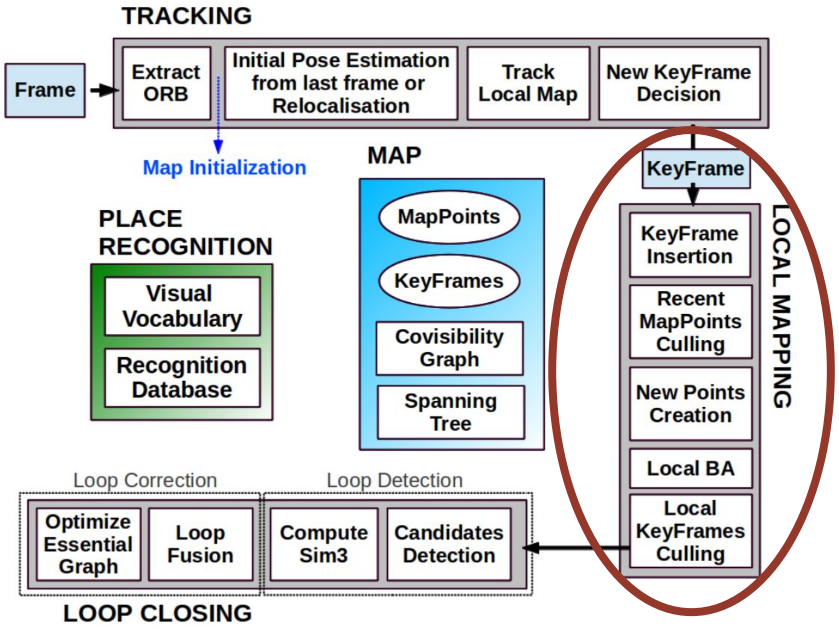


- Key Frame Decision: Important to maintain runtime
  - Too many keyframes requires more compute
  - Too few results in poor localization/mapping performance

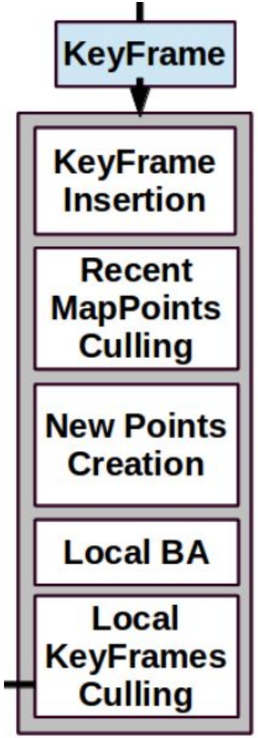
- Decision based on a number of Heuristics
  - Appear to give good performance but no guarantees

Outline:

1. Motivation
2. Definitions
3. Tracking
4. **Local Mapping**
5. Loop Closing

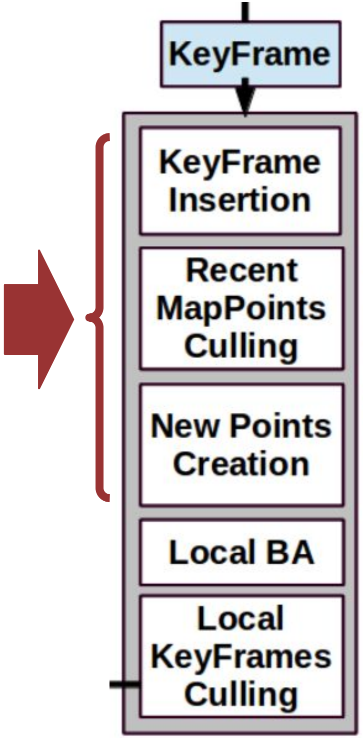


Local Mapping: Goal

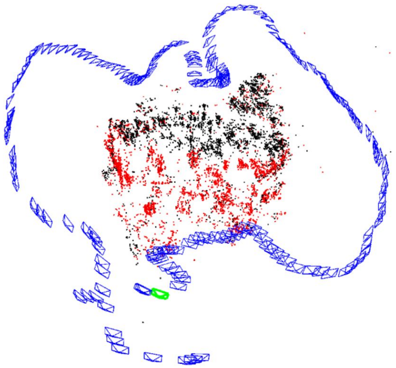


- Given a new **KeyFrame**
  - Grow the map
  - Optimize the map locally
- Occurs at medium rate
  - Defined by keyframe decision heuristics / motion of camera

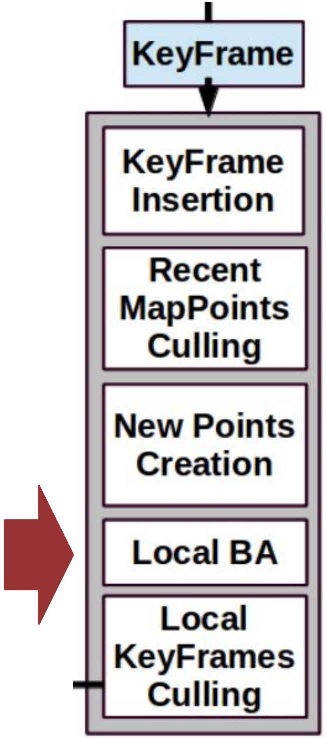
Local Mapping: Adding a Keyframe



- To Insert a New Keyframe
  - For all features in the keyframe
    - If associated with map point
      - Merge with point (Average descriptor/view dir)
    - If NOT associated
      - Add new point to map

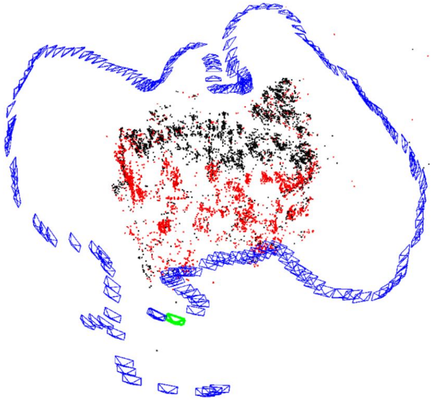


Local Mapping: Local Bundle Adjustment



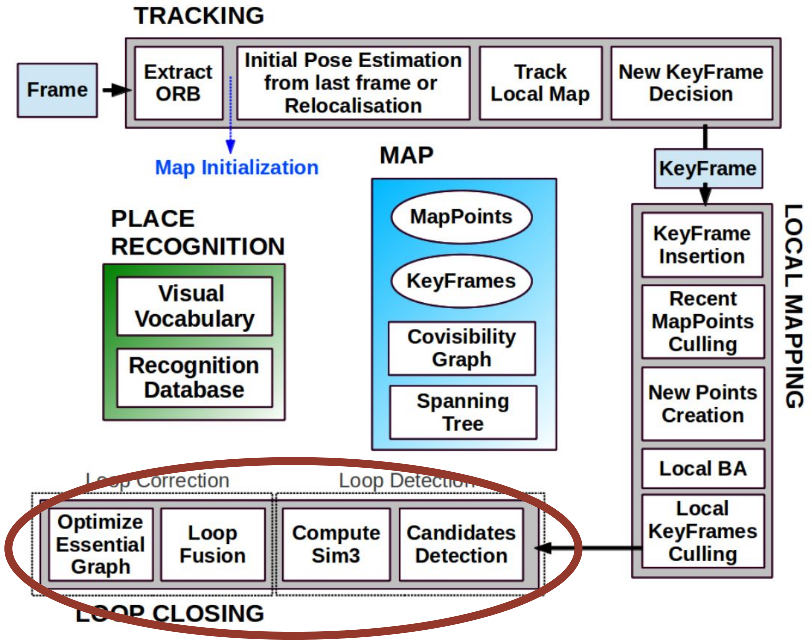
- Preform Bundle Adjustment for Local Map
- Local Map Derived from Co-Visibility Graph
  - All Co-visible keyframes and their neighbors

- Outcome: Red points and associated keyframes are optimized

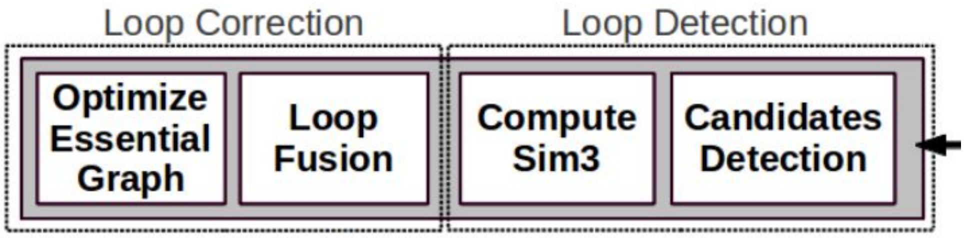


Outline:

1. Motivation
2. Definitions
3. Tracking
4. Local Mapping
5. **Loop Closing**



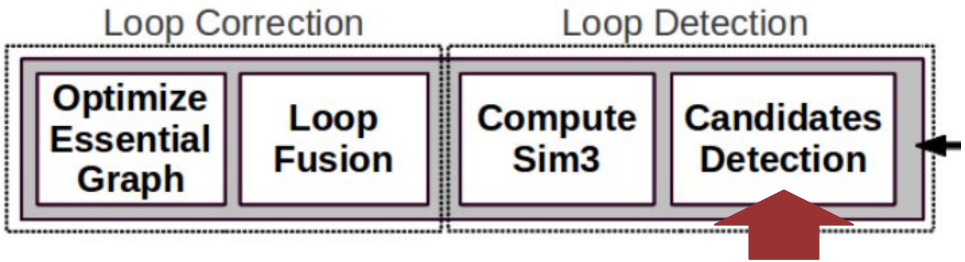
Loop Closures: Goal



- Given a new **KeyFrame**
  - Is it a loop closure?
  - If so, how can we correct for drift

- Occurs at low rate
  - Only when candidate loop-closures are detected

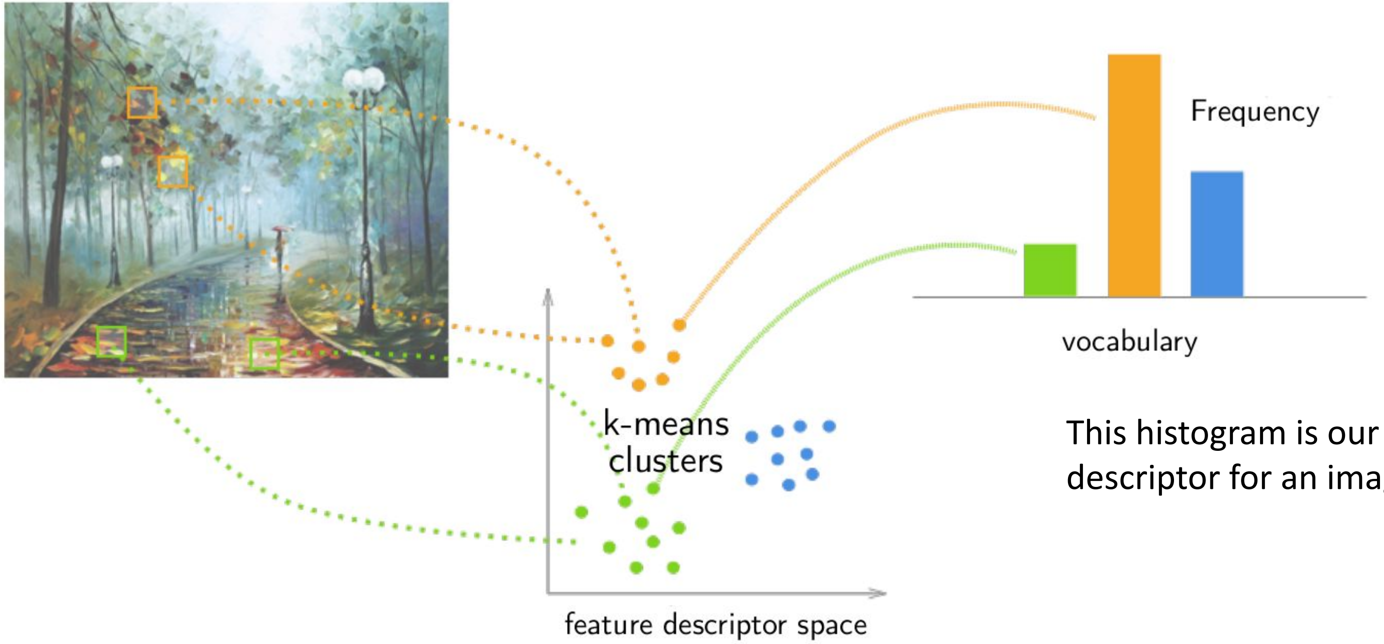
Loop Closures: Candidate Detections



- Based on DBoW2 [1]
  - Hierarchical Bag-of-Words based matching

[1] Gálvez-López, Dorian, and Juan D. Tardos. "Bags of binary words for fast place recognition in image sequences." IEEE Transactions on Robotics 28.5, 2012:1188-1197.

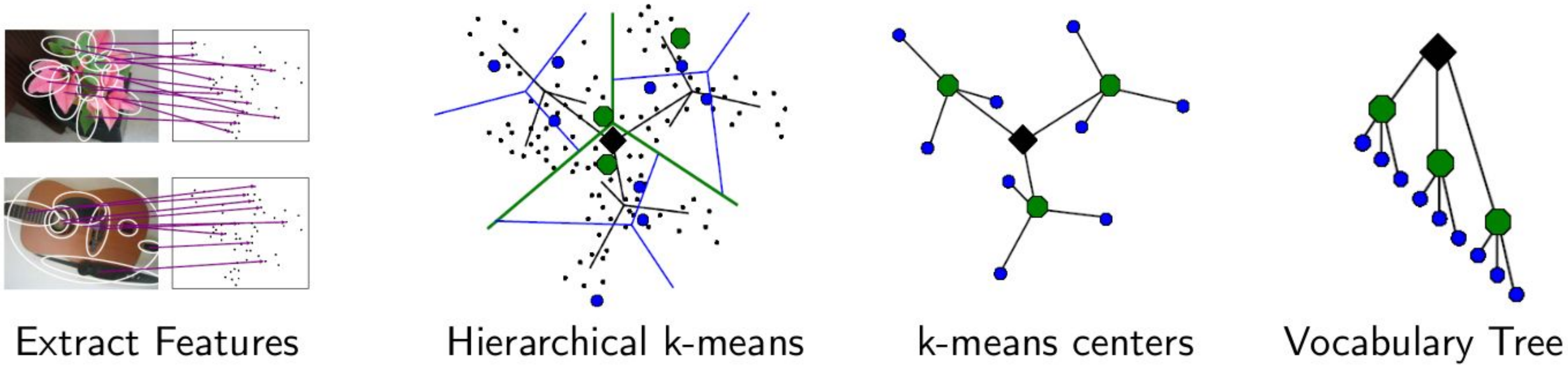
DBoW2: What is a Bag-of-Words?



This histogram is our descriptor for an image

[1] Gálvez-López, Dorian, and Juan D. Tardos. "Bags of binary words for fast place recognition in image sequences." IEEE Transactions on Robotics 28.5, 2012:1188-1197.

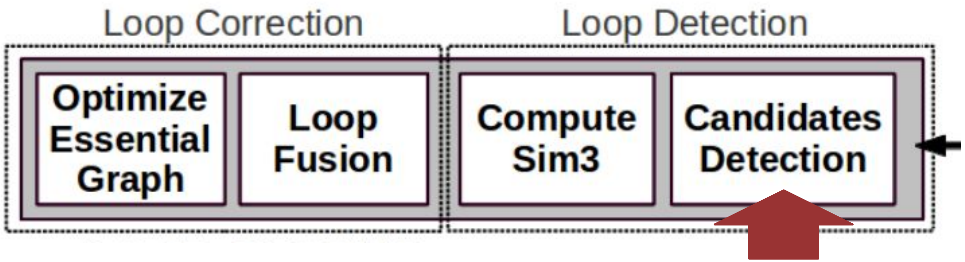
DBoW2: What is hierarchical Bag-of-Words?



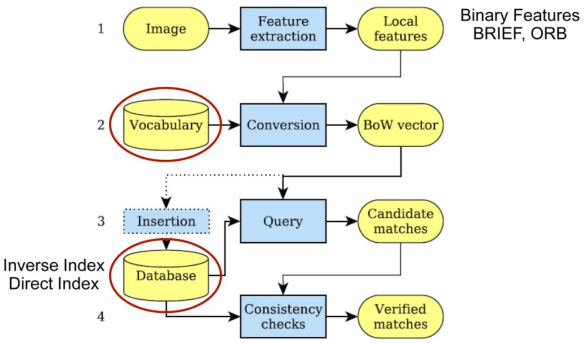
- Cluster the “words” in each cluster to make a hierarchy
- Generates a richer descriptor for each image
- Cluster “Vocabulary” is trained offline

[1] Gálvez-López, Dorian, and Juan D. Tardos. "Bags of binary words for fast place recognition in image sequences." IEEE Transactions on Robotics 28.5, 2012:1188-1197.

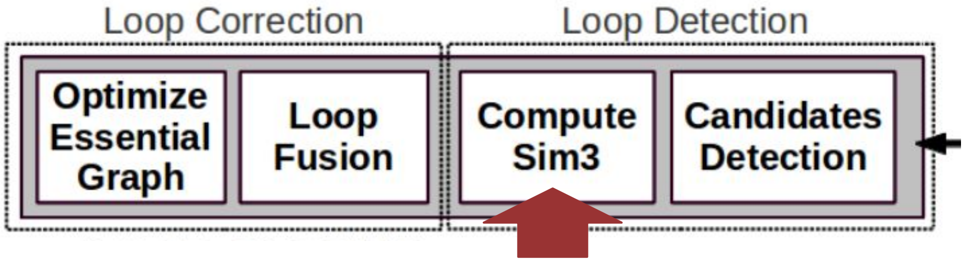
Loop Closures: Candidate Detections



- With an image descriptor we can query database for similar images
  - Excludes local neighbors, and uses some other heuristics in search



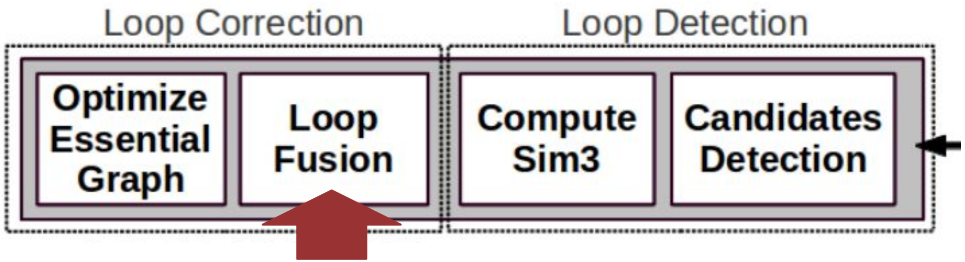
Loop Closures: Computing SIM 3



- If a match is found we want to compute the relative pose
  - Can only do so up to scale (SIM(3))
  - Optimize the following costs (nonlinear)

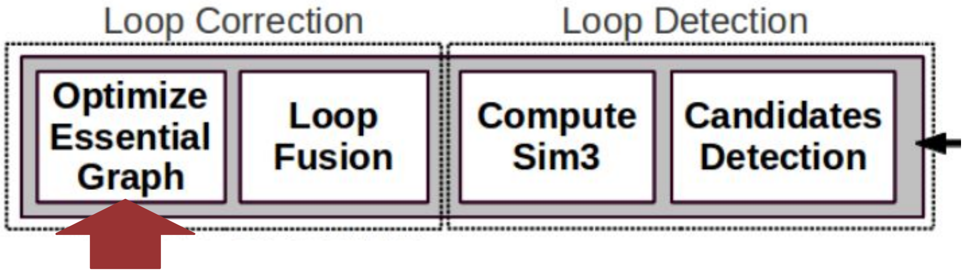
$$\mathbf{e_1} = \mathbf{x}_{1,i} - \pi_1(\mathbf{S}_{12}, \mathbf{X}_{2,j})$$
$$\mathbf{e_2} = \mathbf{x}_{2,j} - \pi_2(\mathbf{S}_{12}^{-1}, \mathbf{X}_{1,i})$$

Loop Closures: Computing SIM 3

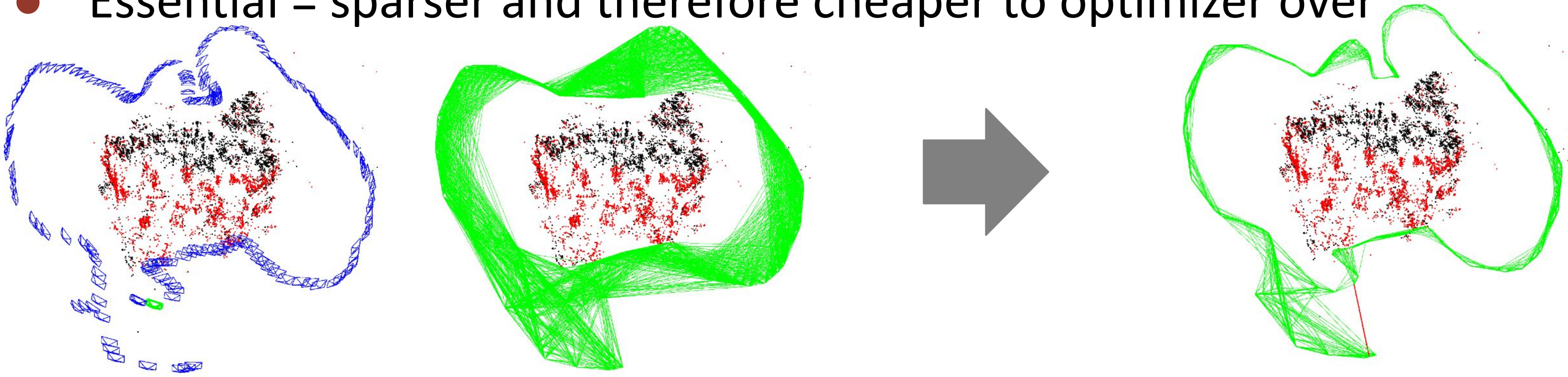


- Simple step to:
  - Fuse duplicate points in the map
  - Add co-visibility edges for the keyframes of fused points

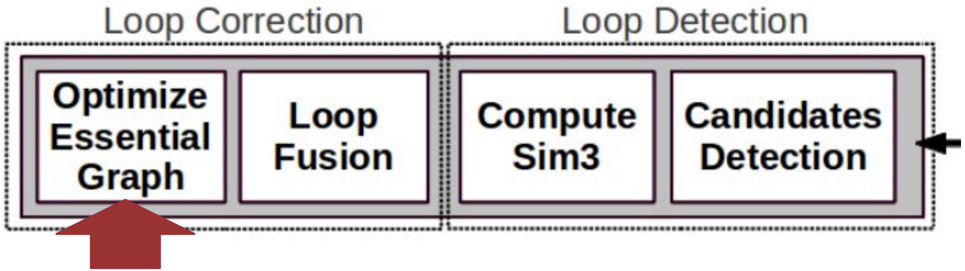
Loop Closures: Optimizing the Graph



- Given current state and map, we first compute the essential graph
- Essential = sparser and therefore cheaper to optimizer over



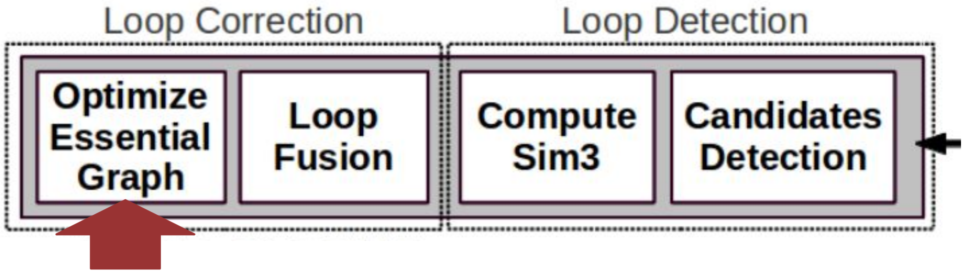
Loop Closures: Optimizing the Graph

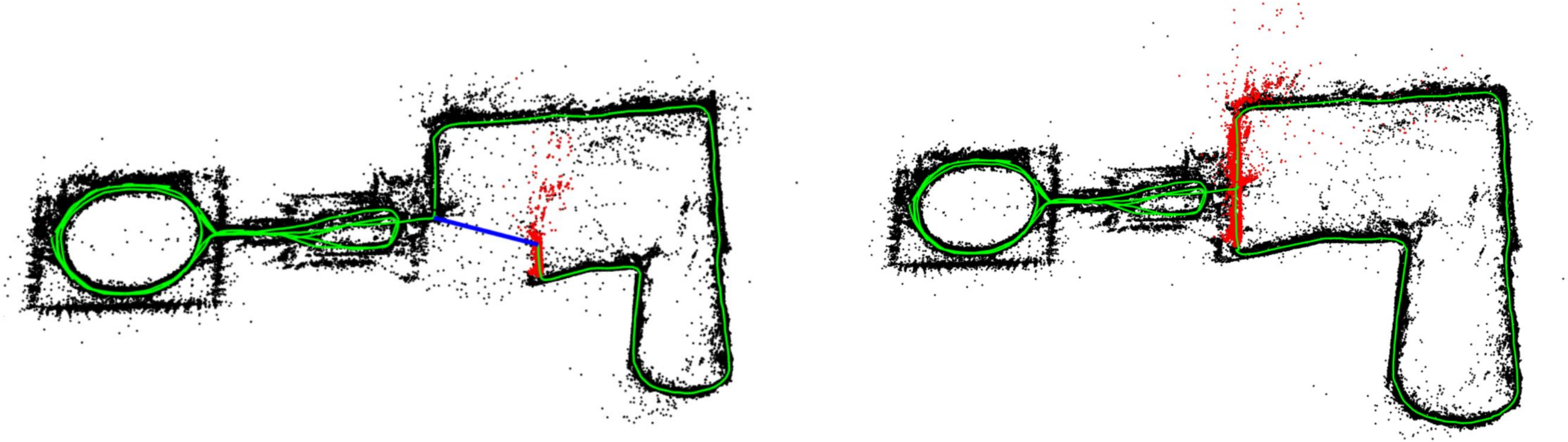


- Essential Graph is optimized using SIM(3) PGO
- Optimizes the following Cost
  - Essentially a variant of HW3 with only (up2scale) relative pose measurements

$$\mathbf{e}_{i,j} = \log_{\mathrm{Sim}(3)}(\mathbf{S}_{ij}\,\mathbf{S}_{jw}\,\mathbf{S}_{iw}^{-1})$$

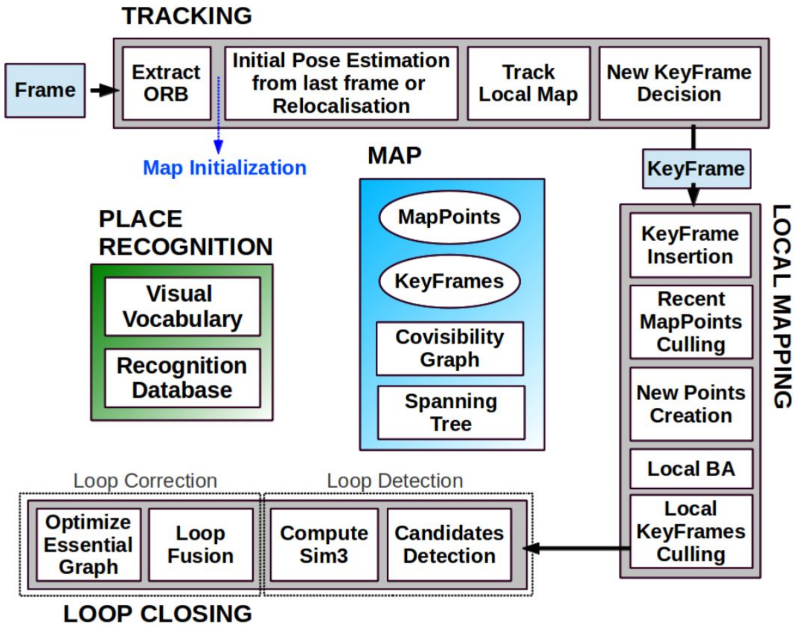
Loop Closures: Optimizing the Graph





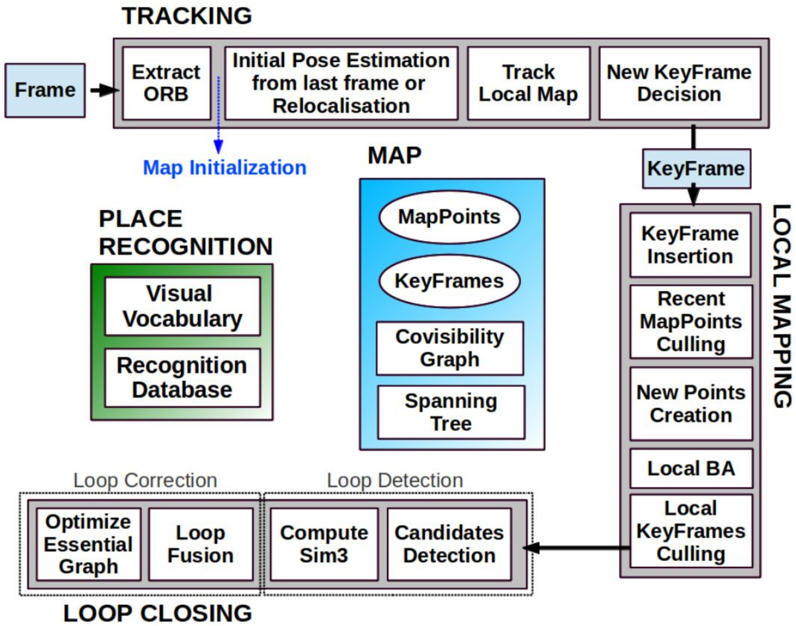
Outline:

1. Motivation
2. Definitions
3. Tracking
4. Local Mapping
5. Loop Closing



ORB-SLAM: Additional Topics

1. Pruning / Culling
2. Initialization
3. Re-Localization
4. Heuristics
5. Robust Estimation



ORB-SLAM: Pruning and Culling

- ORB-SLAM includes an INVOLVED pruning and culling procedure
  - The details are too much to cover in lecture
- The main reason is for tractability: We cannot let the map grow indefinitely over time
- Both keyframes and feature points are pruned

ORB-SLAM: Initialization

- Triangulating first features off of 2 images is hard
  - Planar scenes can cause degenerate solutions
- ORB-SLAM implements two methods
  - 1. Assumes a Planar Scene
  - 2. Assumes a significantly un-planar scene
- Algo uses heuristics to determine which model to use at runtime

ORB-SLAM: Relocalization

- If tracking ever fails i.e.
  - Not enough matches to local map
  - Numerical degeneracies in optimizing
- We can re-localize using our Loop-Closure procedure
  - Minus PGO
- Can still fail. Why?

ORB-SLAM: Heuristics

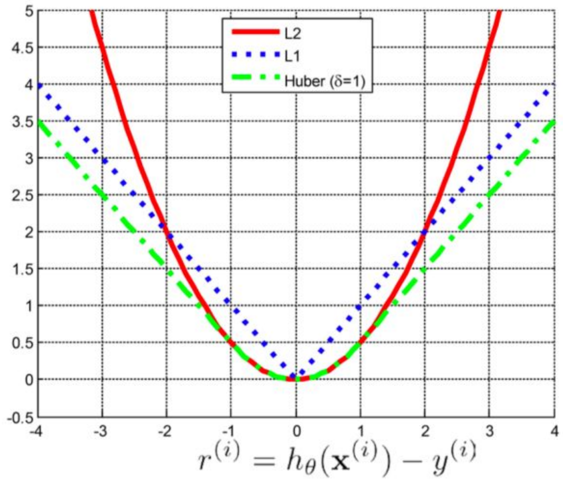
- As you have noticed ORB-SLAM is FULL of heuristics

- I mention this to highlight the challenges of implementing a FULL SLAM system
  - We often need make approximations (PGO vs BA)
  - There are always edge cases to consider (initialization)
  - We need these to run in real-time (pruning)

- SLAM (robotics really) is a balance between correct theory, and practically getting things to run

ORB-SLAM: Robust Estimation

- All non-linear optimizations in ORB-SLAM use Robust Estimators
  - Key Idea: Downweight potential outliers
  - Accomplished with wrapping costs with a “robust kernel”



I will stay on for

Questions?