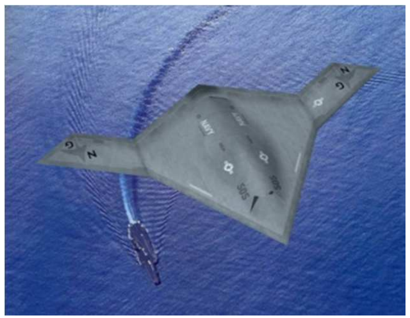
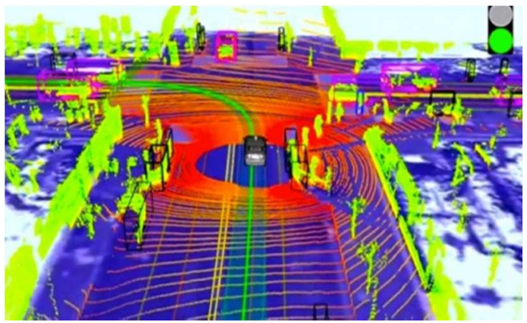
Intelligent Image and Graphics Processing

智能图像图形处理

布树辉 bushuhui@nwpu.edu.cn
http://www.adv-ci.com

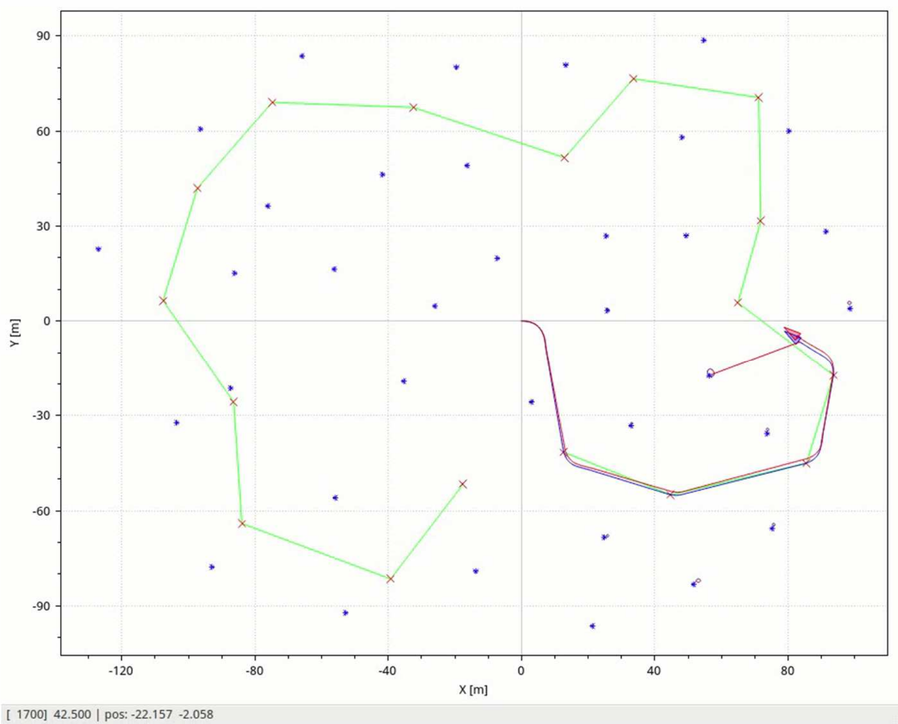




Contents

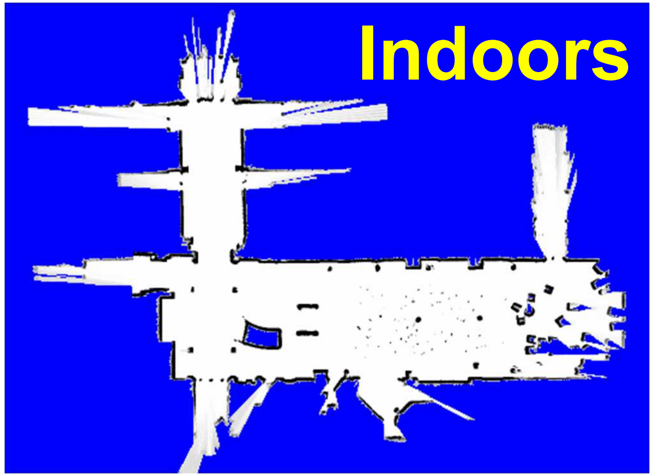
1. SLAM - Application

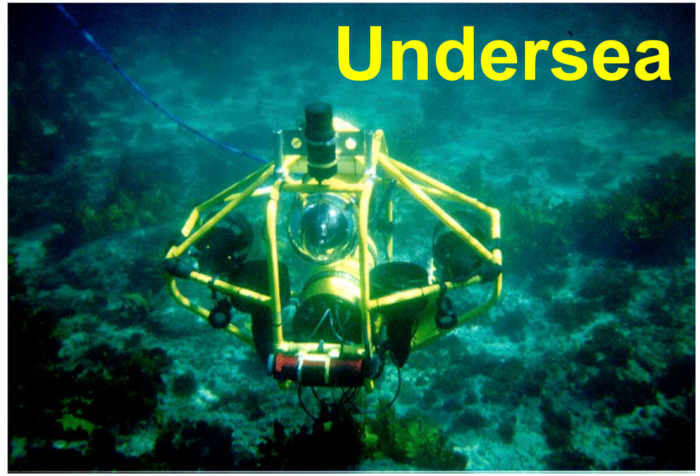
- SLAM: Simultaneous Localization and Mapping
- Applications: Robotics, UAVs, Cars, AR/VR, Smart phones, etc.

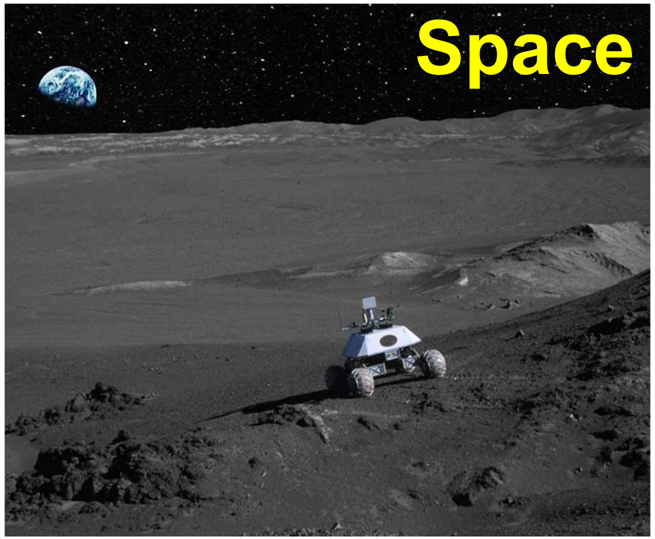


Demo can be downloaded at: http://www.adv-ci.com/blog/source/fastslam-gui/

1. SLAM - Application







1. SLAM - Components

Localization

- Where am I?

Needs accurate map

Needs accurate position

Mapping

- What is around me?

Simultaneously minimization both error

1. SLAM - Sensors

Sensors

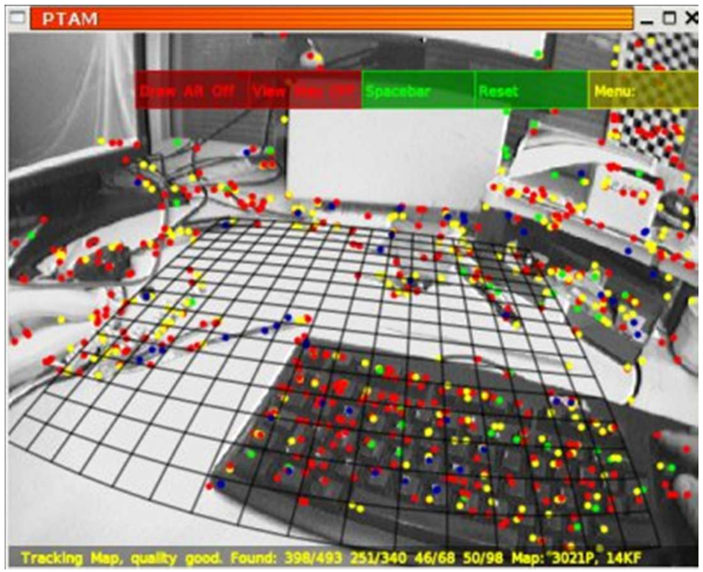
The methods and difficulty of SLAM depend heavily on the equipped sensors.

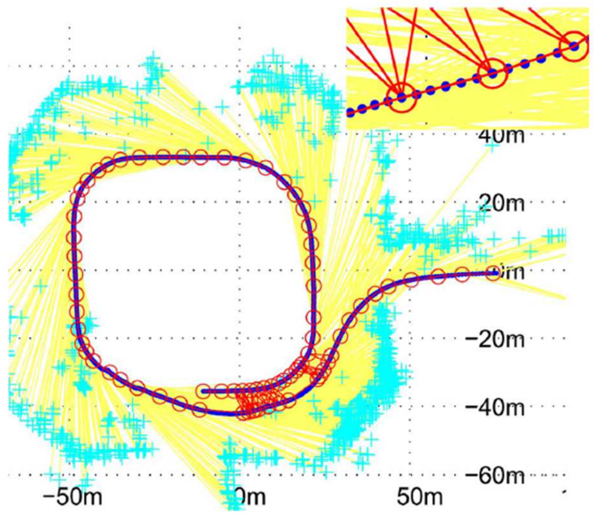
- Lasers
  - Accurate
  - Fast
  - Long history in research
  - Heavy
  - Expensive
- Examples: SICK, Velodyne, Rplidar


- Cameras
  - Cheap
  - Light-weight
  - Rich information
  - High computation cost
  - Work under assumptions
- Categories: monocular, stereo, RGBD


1. SLAM - vSLAM

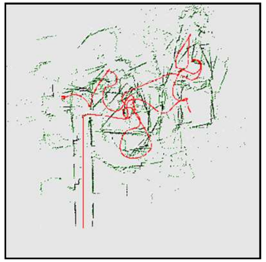
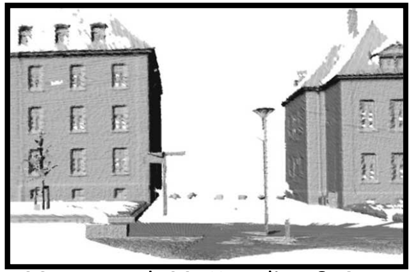
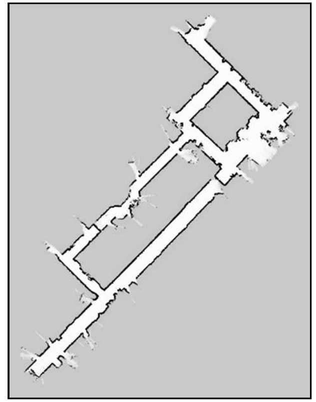
- vSLAM: SLAM that uses cameras as the only or main sensor.
- Active research area in recent decades.
- Considered basically solved in rigid and static environments.





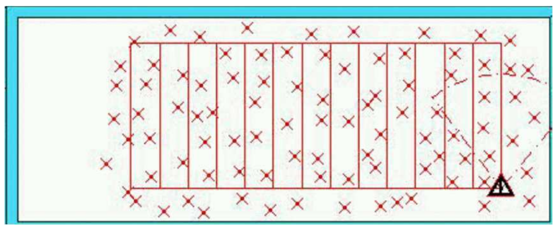
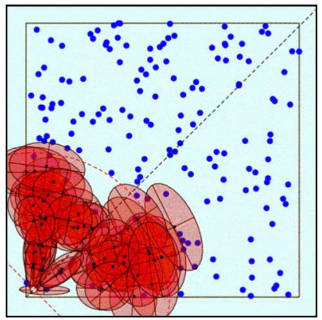
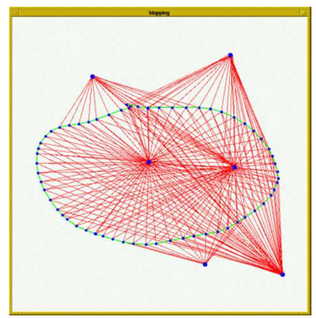
1. SLAM – Map Representation

- Grid maps or scans

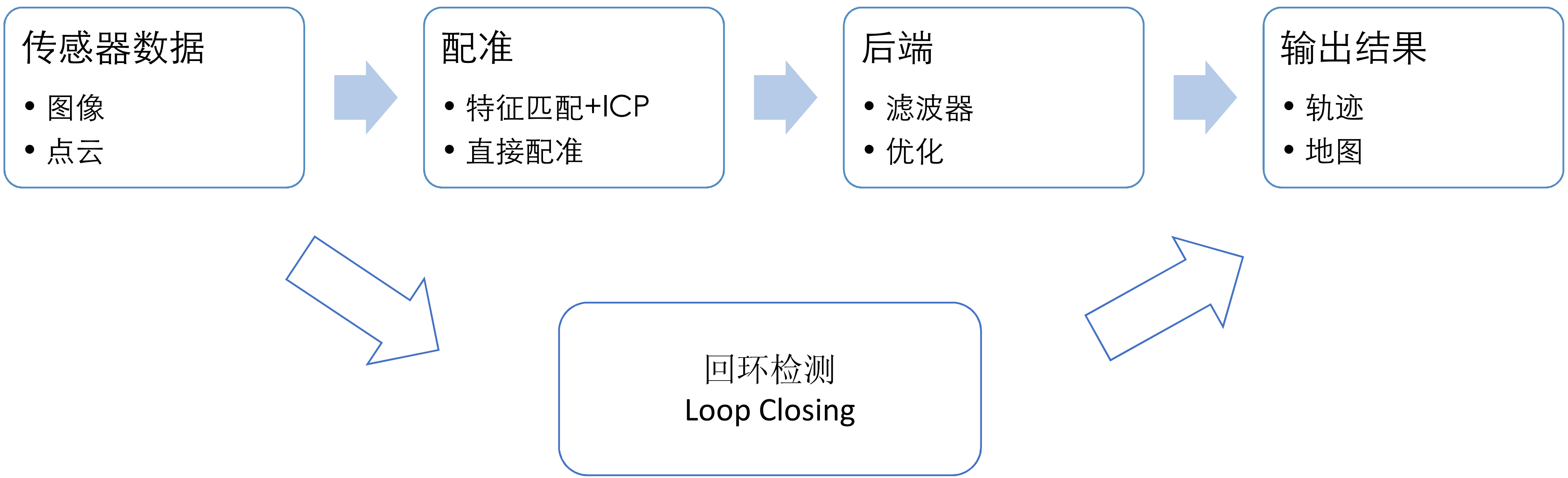
[Lu & Milios, 97; Gutmann, 98: Thrun 98; Burgard, 99; Konolige & Gutmann, 00; Thrun, 00; Arras, 99; Haehnel, 01;…]

- Landmark-based

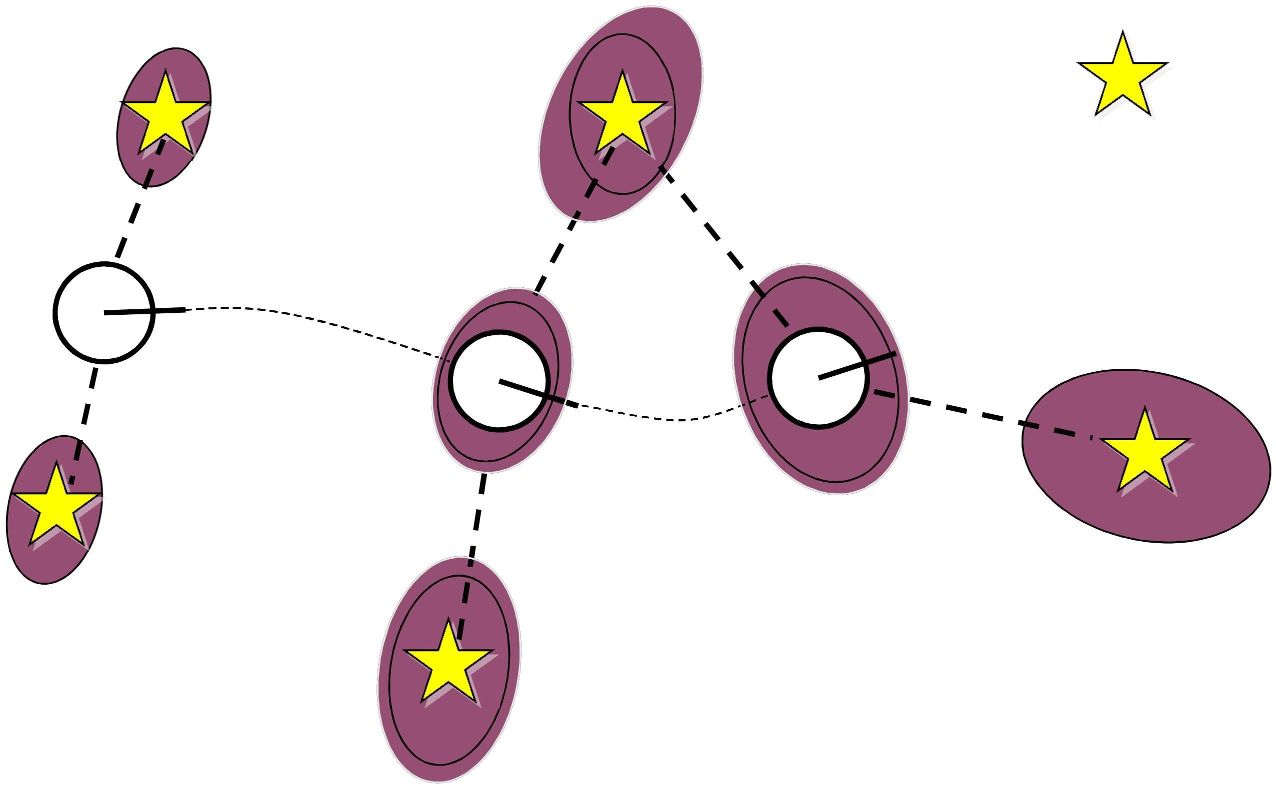
[Leonard et al., 98; Castelanos et al., 99: Dissanayake et al., 2001; Montemerlo et al., 2002;…

1. SLAM Pipeline



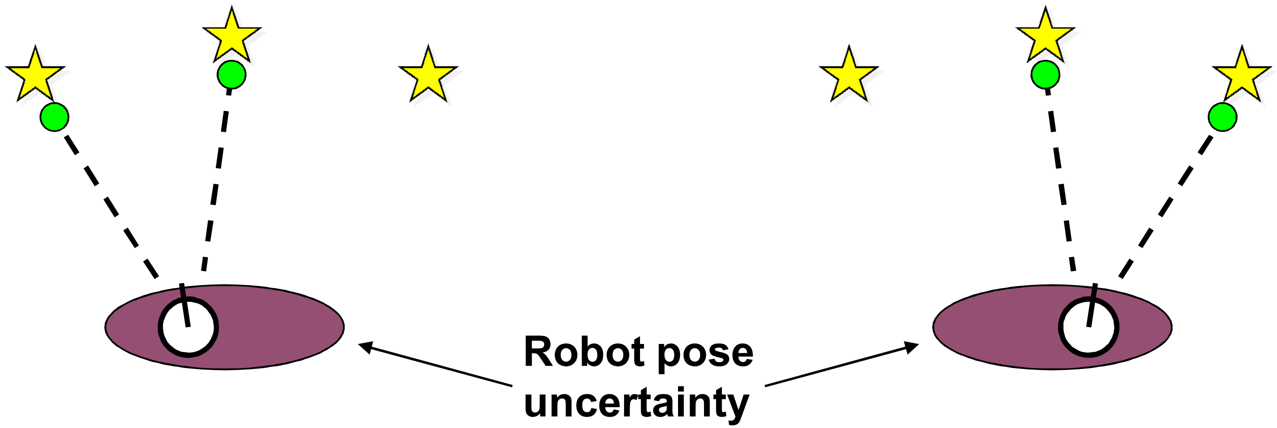
1. SLAM is a hard problem!

SLAM: robot path and map are both **unknown**



Robot path error correlates errors in the map

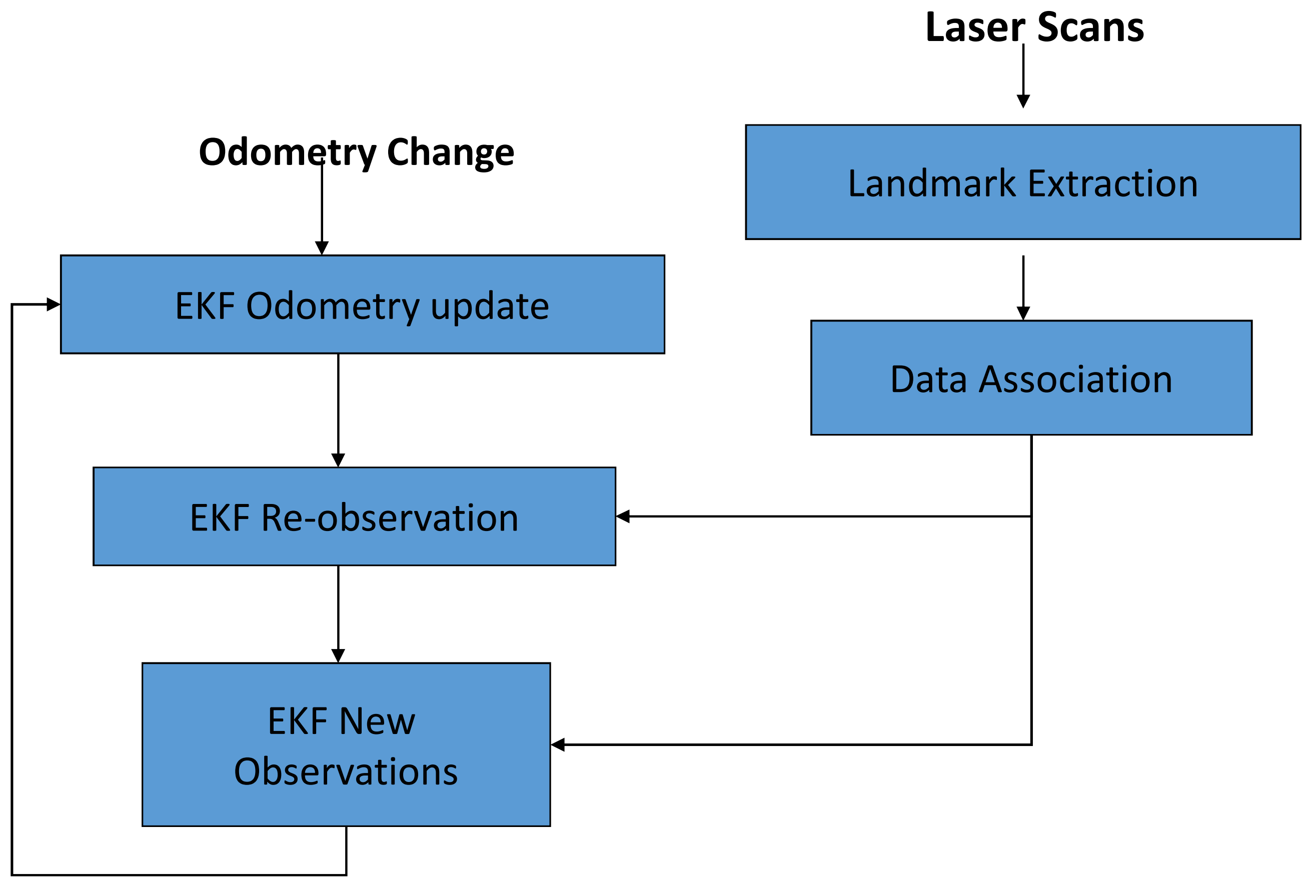
1. SLAM is a hard problem!



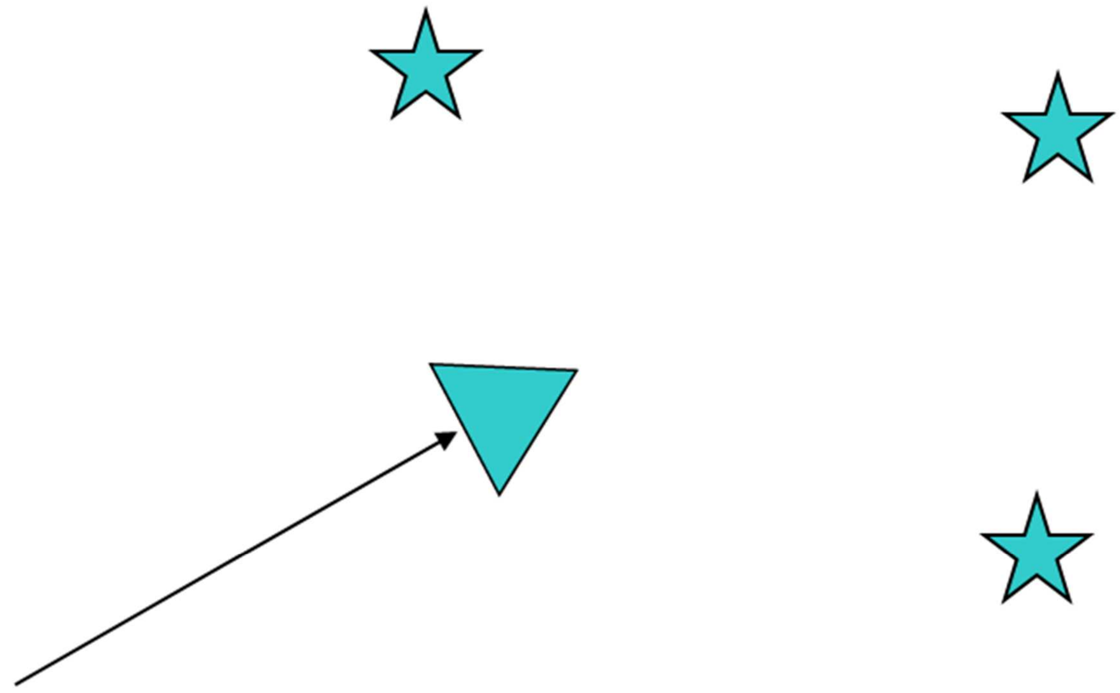
- In the real world, the mapping between observations and landmarks is unknown
- Picking wrong data associations can have catastrophic consequences
- Pose error correlates data associations

Probabilistic Way – EKF SLAM

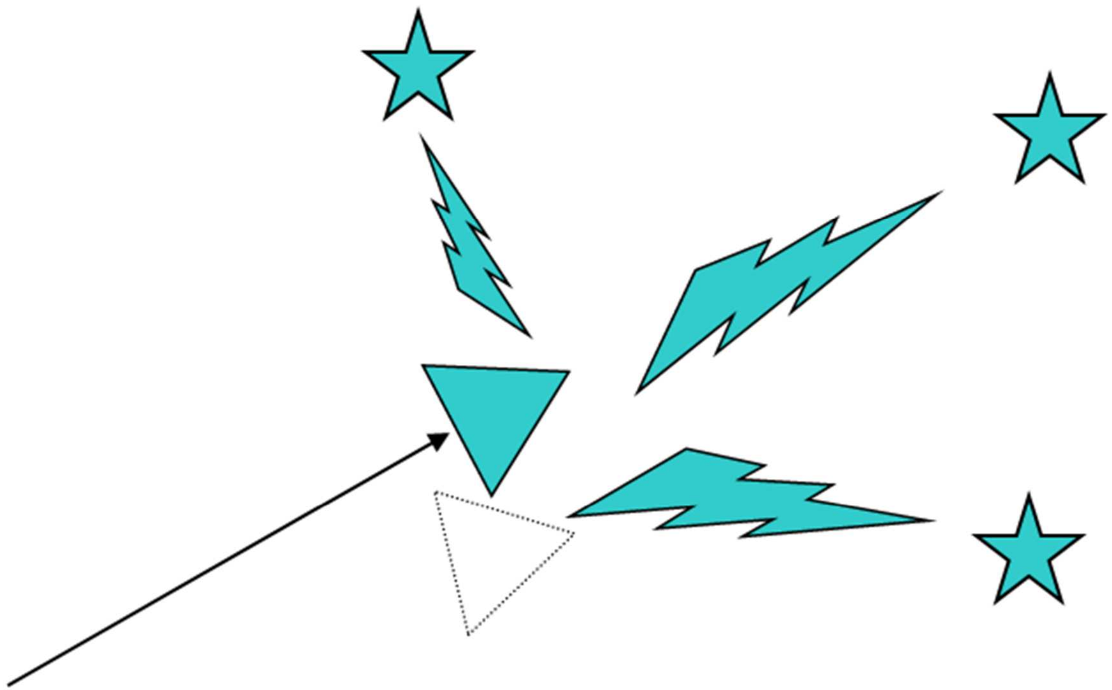
Probabilistic Way



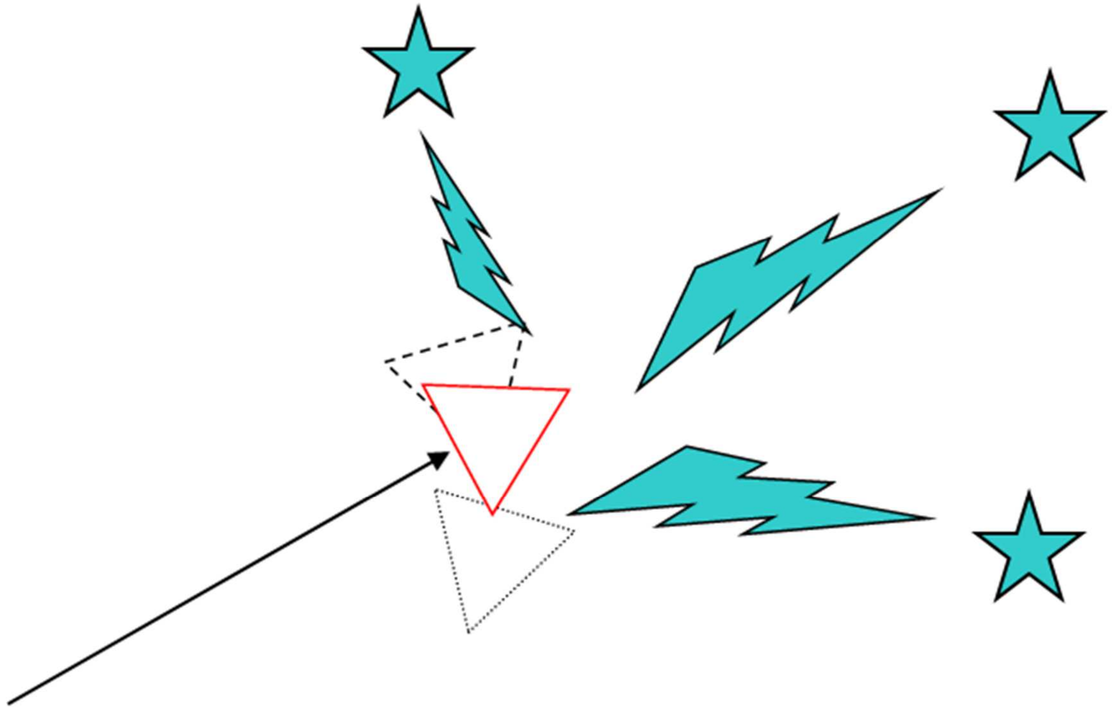
Probabilistic Way



Probabilistic Way



Probabilistic Way



Probabilistic Way - Laser & Odometry data

- Laser data is the reading obtained from the scan
- The goal of the odometry data is to provide an approximate position of the robot
- The difficult part about the odometry data and the laser data is to get the timing right.

Probabilistic Way - Landmarks

- Landmarks are features which can easily be re-observed and distinguished from the environment.
- These are used by the robot to find out where it is (to localize itself).

  - Landmarks should be easily re-observable.
  - Individual landmarks should be distinguishable from each other.
  - Landmarks should be plentiful in the environment.
  - Landmarks should be stationary.

Probabilistic Way - Terminology

- Robot State (or pose): $x_t = [x, y, \theta]$
  - Position and heading
  - $x_{1:t} = \{x_1, \ldots, x_t\}$
- Robot Controls: u_t
  - Robot motion and manipulation
  - $u_{1:t} = \{u_1, \ldots, u_t\}$
- Sensor Measurements: z_t
  - Range scans, images, etc.
  - $z_{1:t} = \{z_1, \ldots, z_t\}$
- Landmark or Map: $m_i \;\; or \;\; l_i$
  - Landmarks or Map

$$m = \{m_1, \ldots, m_n\} \; or \; l = \{l_1, \ldots, l_n\}$$

Probabilistic Way - Terminology

- Observation model: $P(z_t \mid x_t)$ or $P(z_t \mid x_t, m)$
  - The probability of a measurement z_t given that the robot is at position x_t *and map m*.
- Motion Model: $P(x_t \mid x_{t-1}, u_t)$
  - The posterior probability that action u_t carries the robot from x_{t-1} to x_t.

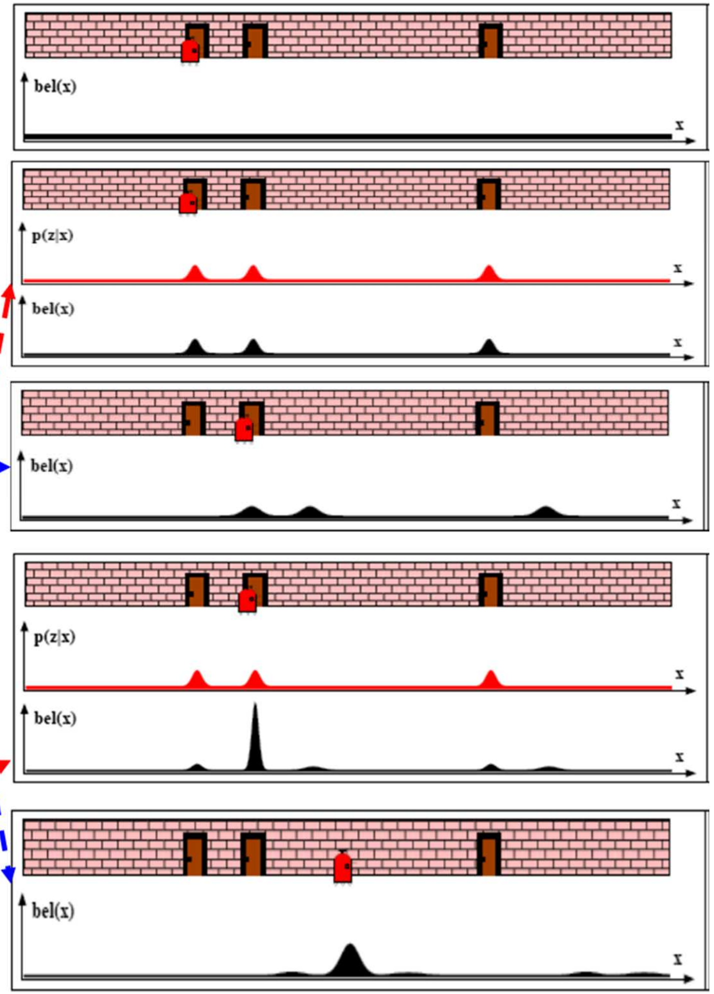
Probabilistic Way - Bayes Filter

- **Prediction**

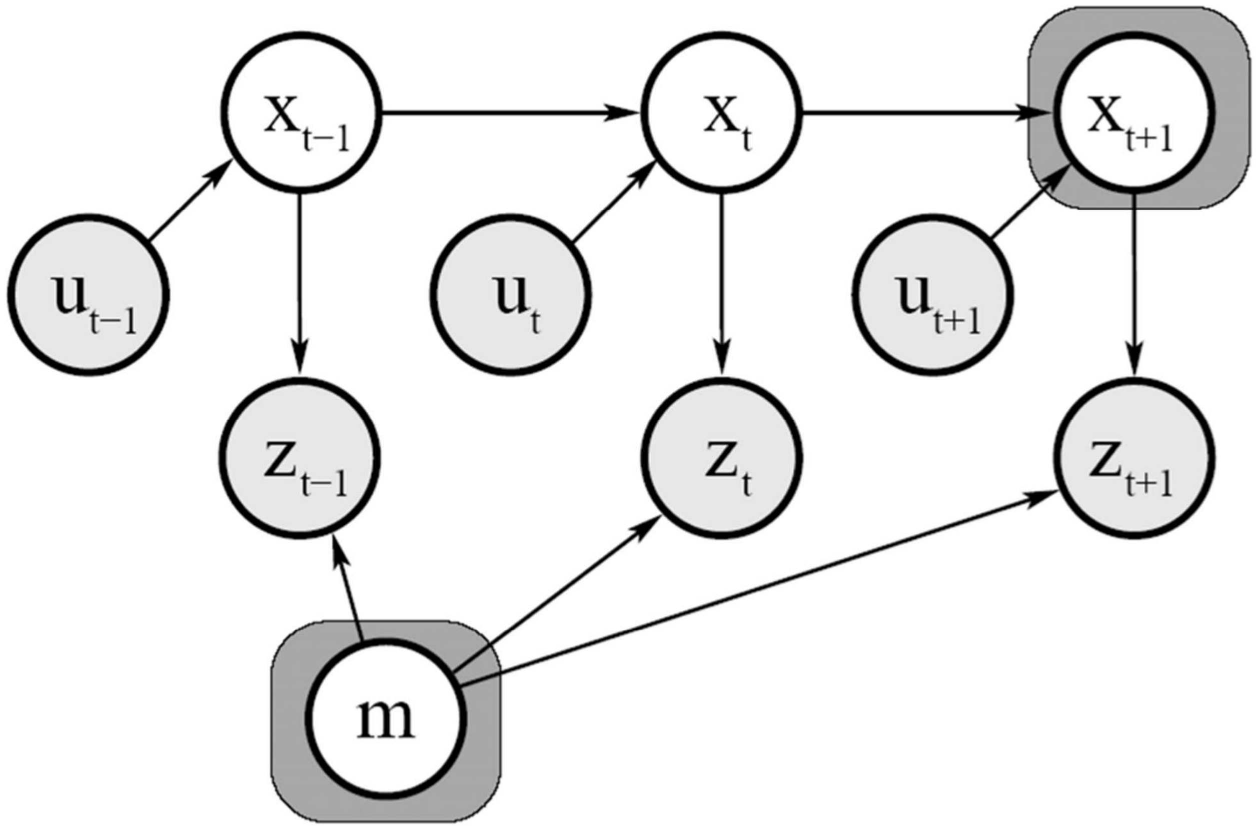
$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$$

- **Update**

$$bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$$

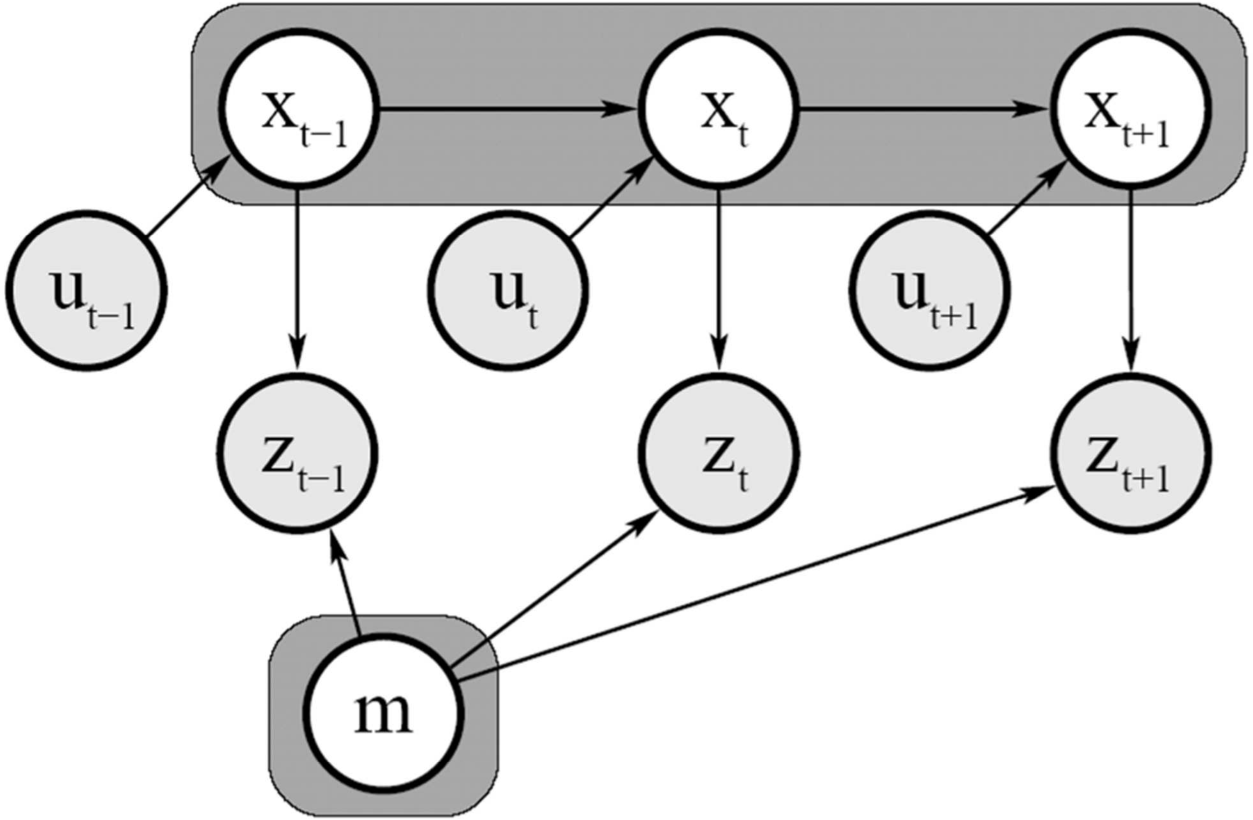


Probabilistic Way – Graphical Model of Online SLAM



$$p(x_t, m \mid z_{1:t}, u_{1:t}) = \int\int \ldots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \; dx_1 \, dx_2 \ldots dx_{t-1}$$

Probabilistic Way – Graphical Model of Full SLAM



$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$$

Probabilistic Way – Scan Matching

Maximize the likelihood of the i-th pose and map relative to the (i-1)-th pose and map.

$$\hat{x}_t = \underset{x_t}{\operatorname{argmax}} \left\{ p(z_t \mid x_t, \hat{m}^{[t-1]}) \cdot p(x_t \mid u_{t-1}, \hat{x}_{t-1}) \right\}$$

current measurement

robot motion

map constructed so far

Calculate the map $\hat{m}^{[t]}$ according to "mapping with known poses" based on the poses and observations.

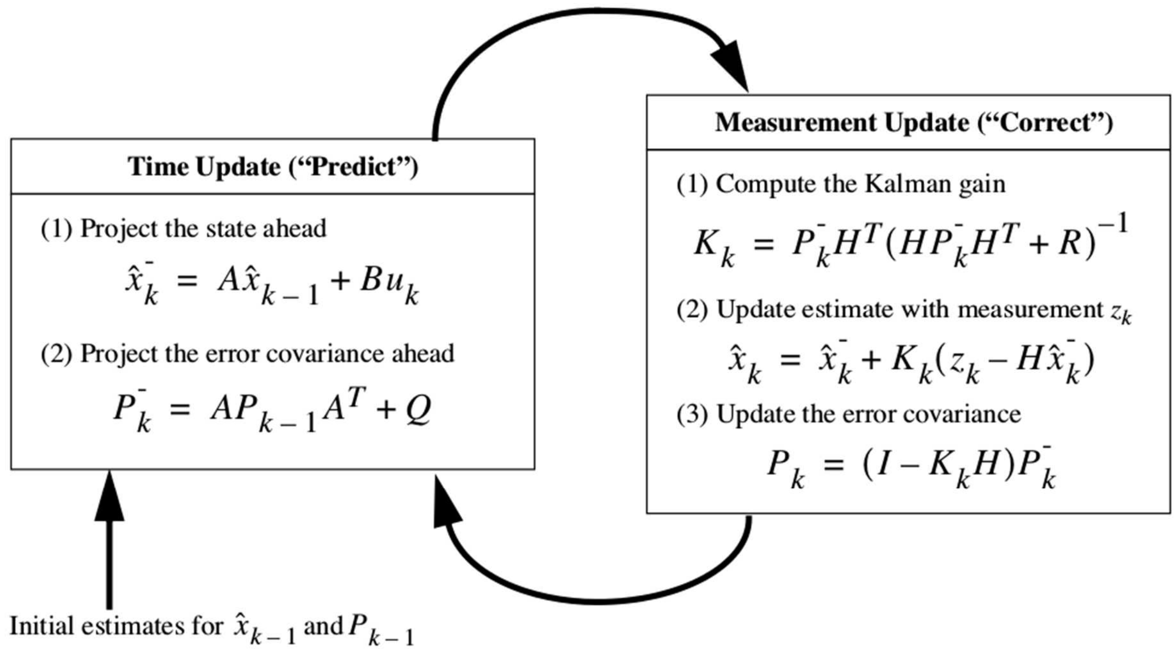
Probabilistic Way – Kalman Filter

1. Algorithm **Kalman_filter**(μ_{t-1}, Σ_{t-1}, u_t, z_t):

2. Prediction:
3. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
4. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

5. Correction:
6. $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
7. $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
8. $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$

9. Return μ_t, Σ_t



Probabilistic Way – EKF-SLAM

- Map with N landmarks:(3+2N)-dimensional Gaussian

$$
Bel(x_t, m_t) = \left\langle \begin{pmatrix} x \\ y \\ \theta \\ l_1 \\ l_2 \\ \vdots \\ l_N \end{pmatrix}, \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xl_1} & \sigma_{xl_2} & \cdots & \sigma_{xl_N} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} & \sigma_{yl_1} & \sigma_{yl_2} & \cdots & \sigma_{yl_N} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 & \sigma_{\theta l_1} & \sigma_{\theta l_2} & \cdots & \sigma_{\theta l_N} \\ \sigma_{xl_1} & \sigma_{yl_1} & \sigma_{\theta l_1} & \sigma_{l_1}^2 & \sigma_{l_1 l_2} & \cdots & \sigma_{l_1 l_N} \\ \sigma_{xl_2} & \sigma_{yl_2} & \sigma_{\theta l_2} & \sigma_{l_1 l_2} & \sigma_{l_2}^2 & \cdots & \sigma_{l_2 l_N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{xl_N} & \sigma_{yl_N} & \sigma_{\theta l_N} & \sigma_{l_1 l_N} & \sigma_{l_2 l_N} & \cdots & \sigma_{l_N}^2 \end{pmatrix} \right\rangle
$$

- Can handle hundreds of dimensions

Algorithm EKF_SLAM_known_correspondences$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t)$:

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 1 & \underbrace{0\cdots0}_{3N} \end{pmatrix}$$

$$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

$$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & \frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

$$\bar{\Sigma}_t = G_t\,\Sigma_{t-1}\,G_t^T + F_x^T\,R_t\,F_x$$

$$Q_t = \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix}$$

for all observed features $z_t^i = (r_t^i\ \phi_t^i\ s_t^i)^T$ do

$j = c_t^i$

if landmark j *never seen before*

$$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \\ \bar{\mu}_{j,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + \begin{pmatrix} r_t^i\cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t^i\sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$$

endif

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{j,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{j,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

$$q = \delta^T\delta$$

$$\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \operatorname{atan2}(\delta_y, \delta_x) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{j,s} \end{pmatrix}$$

$$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 1 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 0 & \underbrace{0\cdots0}_{3j-3} & 0 & 0 & 1 & \underbrace{0\cdots0}_{3N-3j} \end{pmatrix}$$

$$H_t^i = \frac{1}{q}\begin{pmatrix} \sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & -\sqrt{q}\delta_x & \sqrt{q}\delta_y & 0 \\ \delta_y & \delta_x & -1 & -\delta_y & -\delta_x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} F_{x,j}$$

$$K_t^i = \bar{\Sigma}_t\,H_t^{iT}(H_t^i\,\bar{\Sigma}_t\,H_t^{iT} + Q_t)^{-1}$$

$$\bar{\mu}_t = \bar{\mu}_t + K_t^i(z_t^i - \hat{z}_t^i)$$

$$\bar{\Sigma}_t = (I - K_t^i\,H_t^i)\,\bar{\Sigma}_t$$

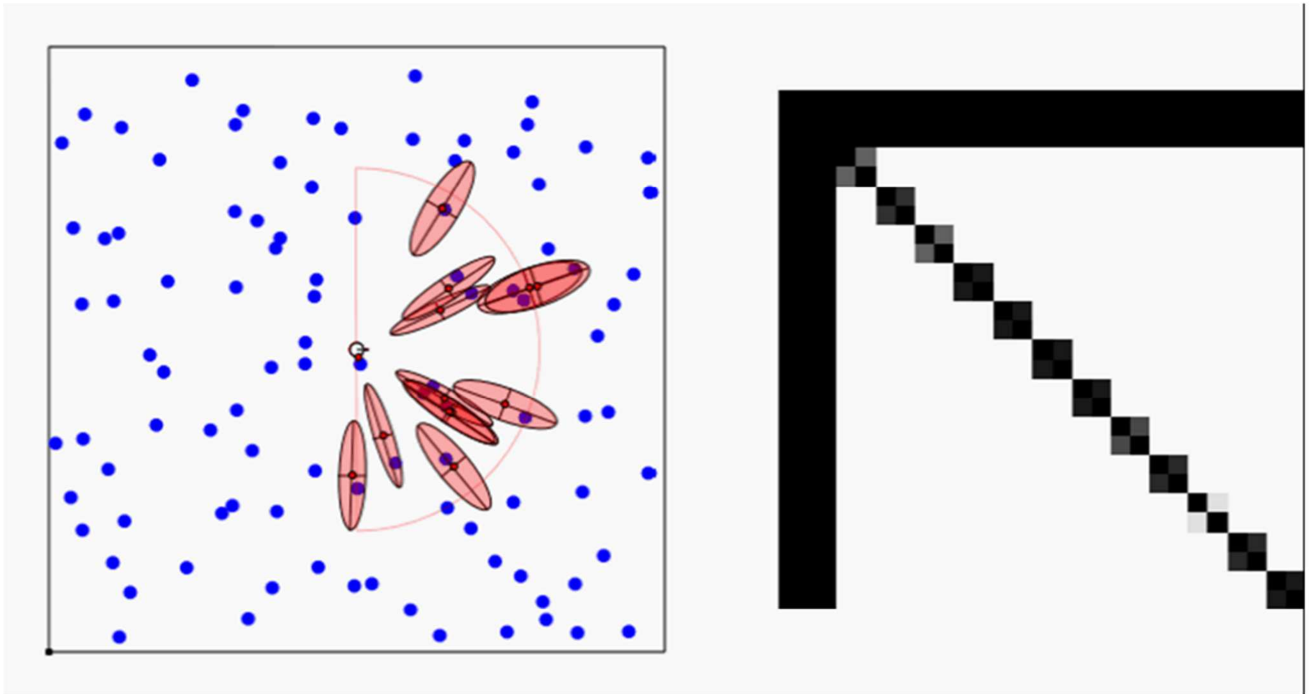
endfor

$\mu_t = \bar{\mu}_t$

$\Sigma_t = \bar{\Sigma}_t$

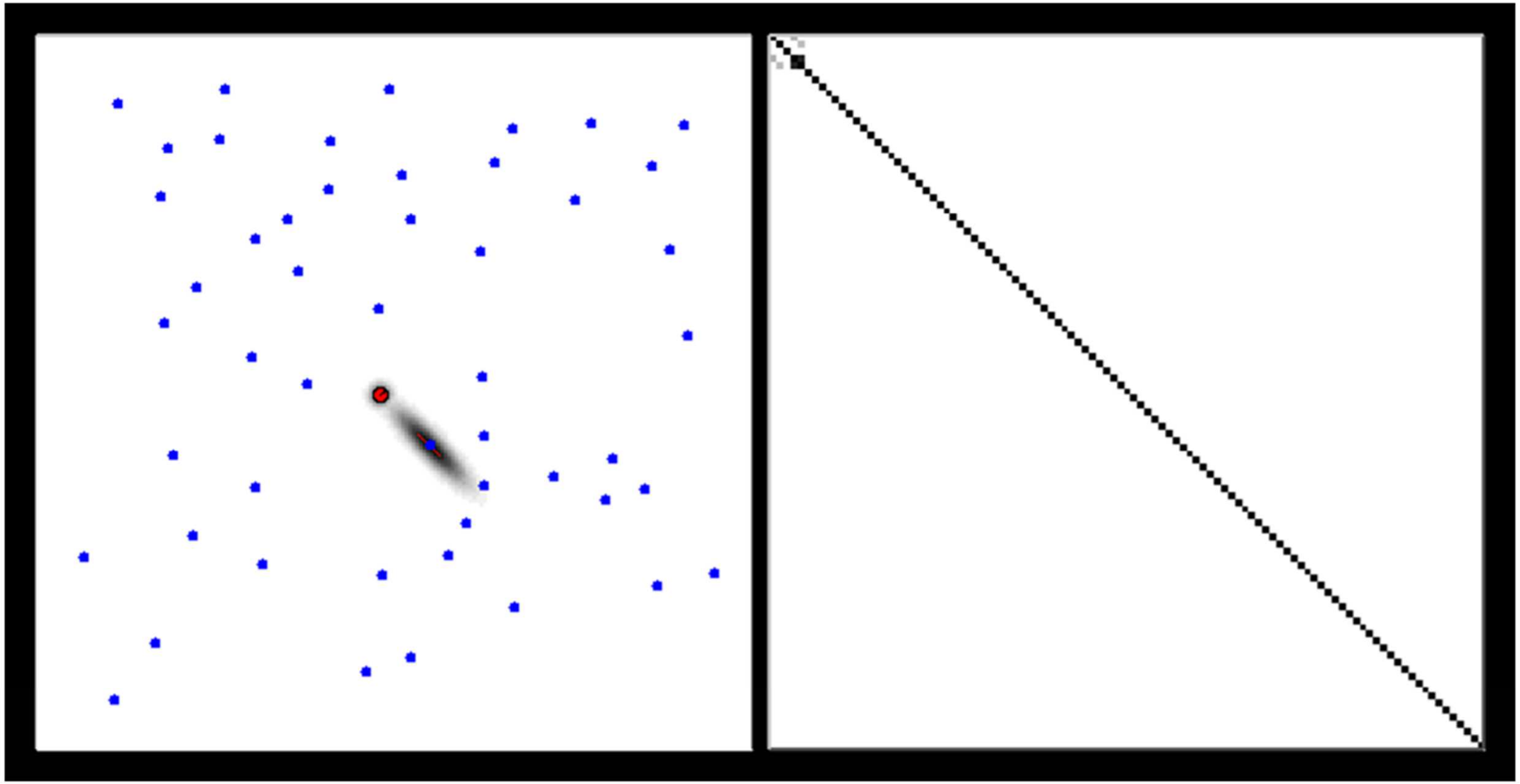
return μ_t, Σ_t

Probabilistic Way – EKF-SLAM



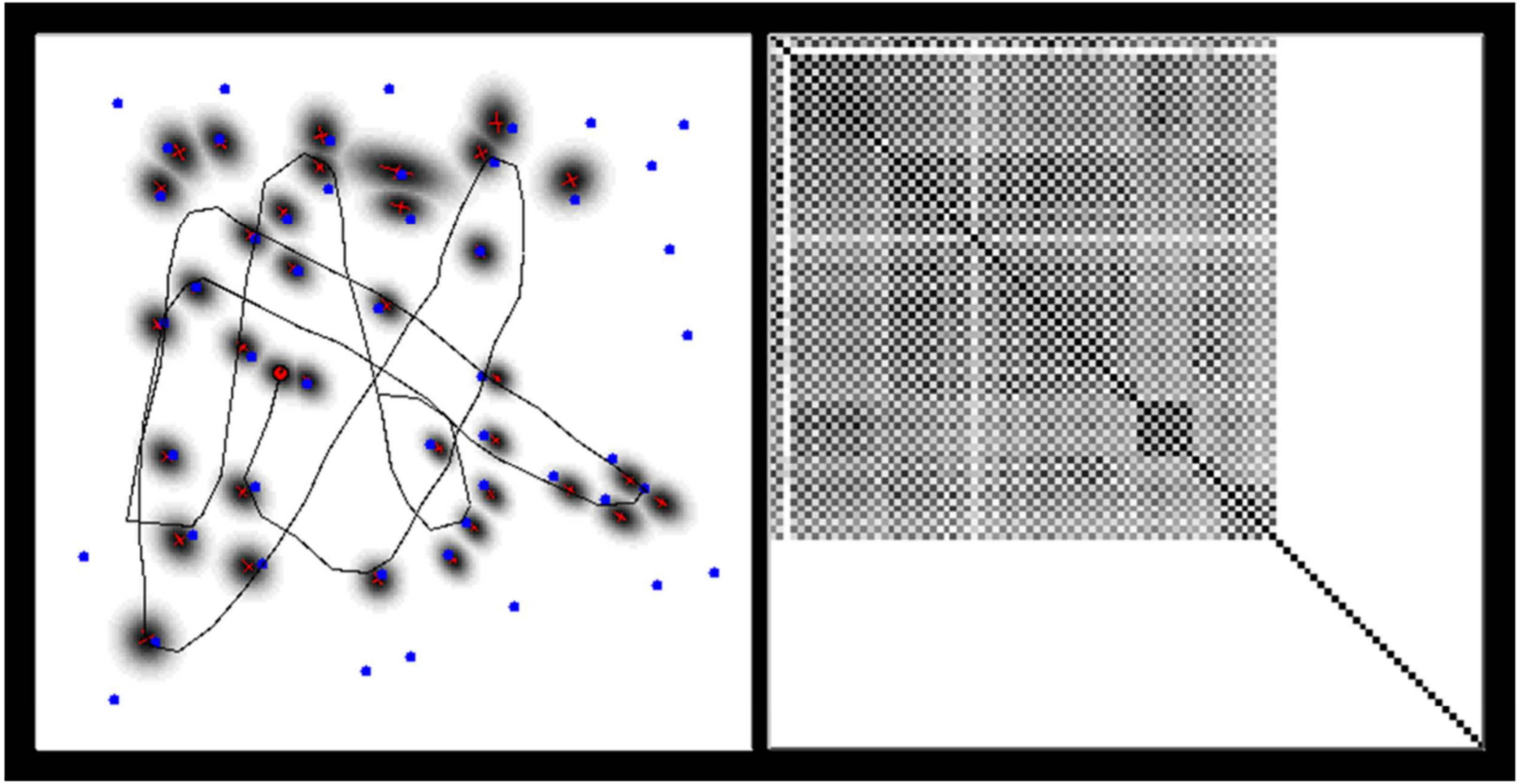
- Approximate the SLAM posterior with a high-dimensional Gaussian [Smith & Cheesman, 1986] ...
- Single hypothesis data association

Probabilistic Way – EKF-SLAM



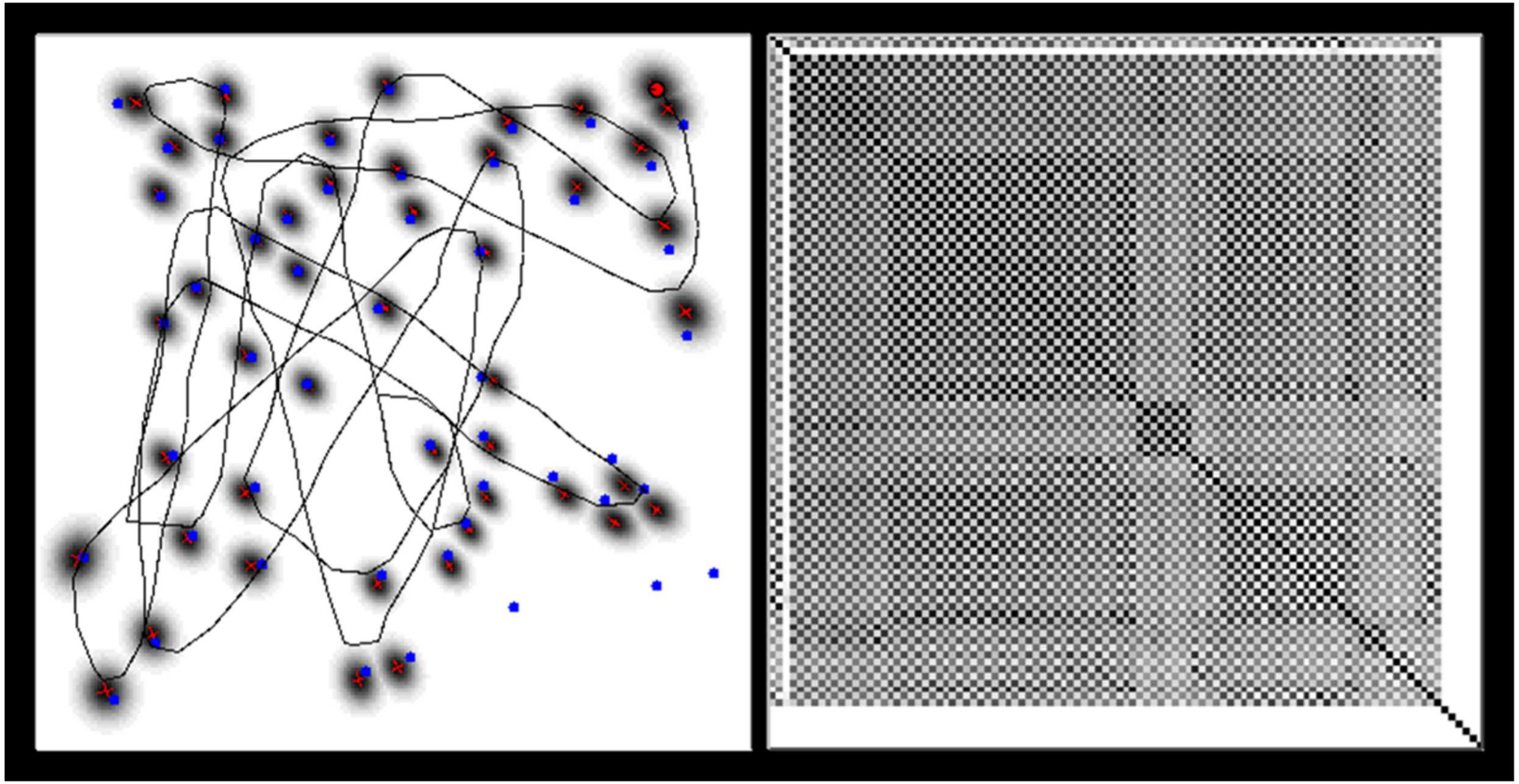
Map Correlation matrix

Probabilistic Way – EKF-SLAM



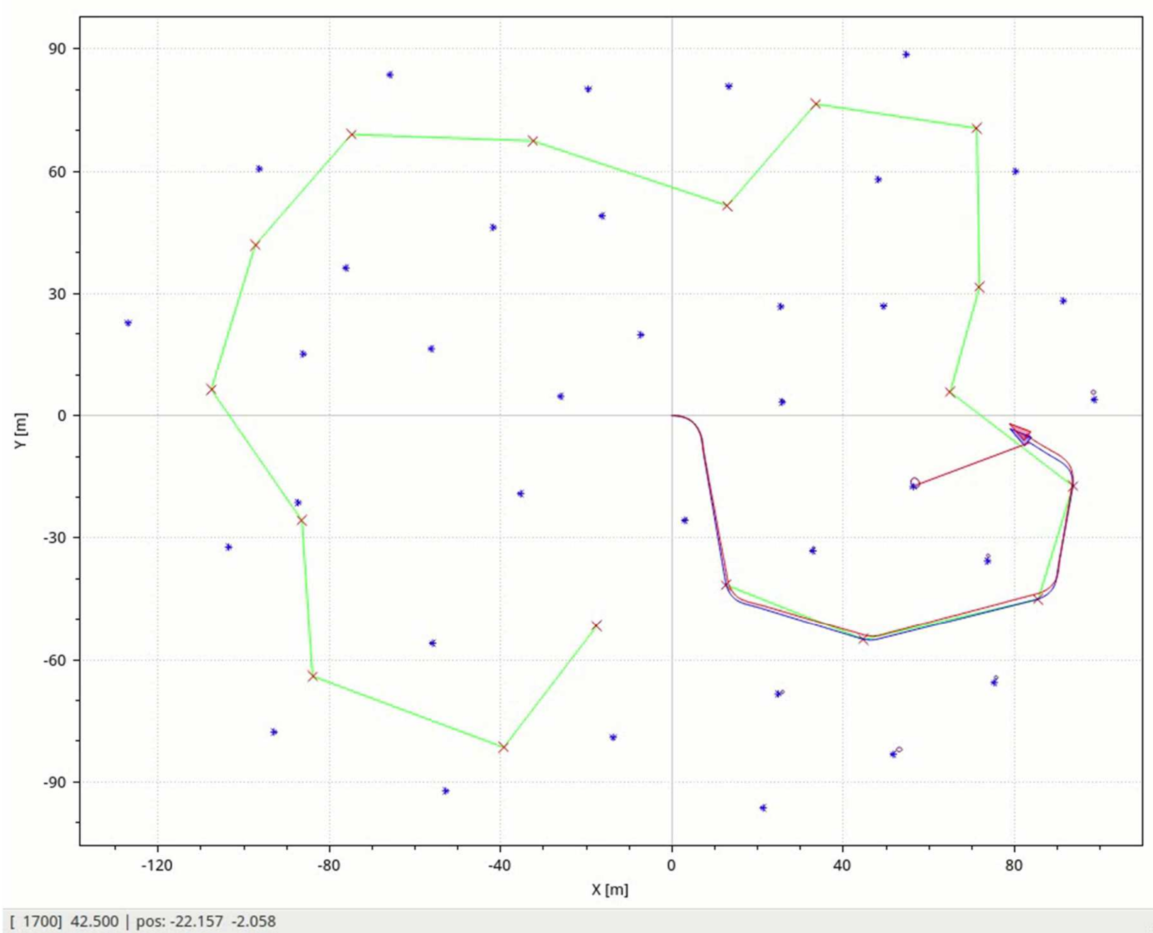
Map Correlation matrix

Probabilistic Way – EKF-SLAM



Map Correlation matrix

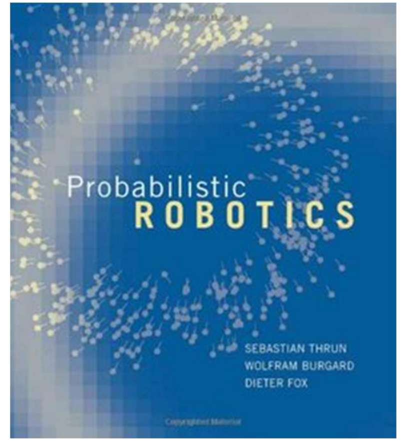
Probabilistic Way – Demo



Demo can be downloaded at: http://www.adv-ci.com/blog/source/fastslam-gui/

Probabilistic Way – Summary

- Quadratic in the number of landmarks: $O(n^2)$
- Convergence results for the linear case.
- Can diverge if nonlinearities are large!
- Has been applied successfully in large-scale environments.
- Approximations reduce the computational complexity.

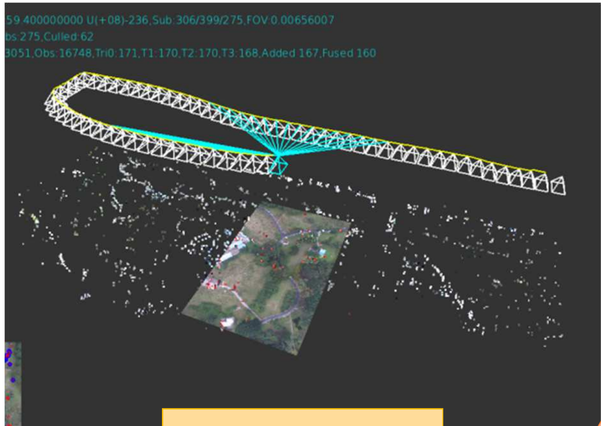


Limitations:

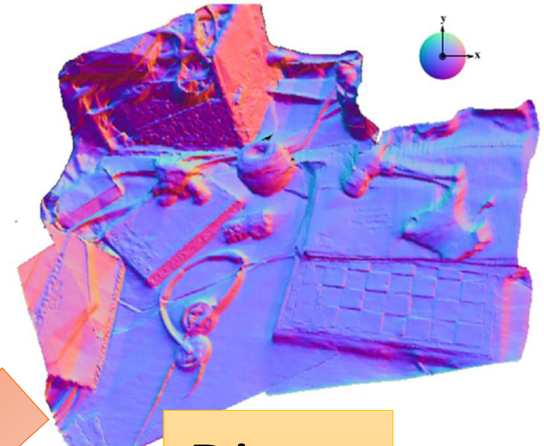
- Computational efficiency is worse then bundle-based method
- Can not handle large number of observations

Visual SLAM

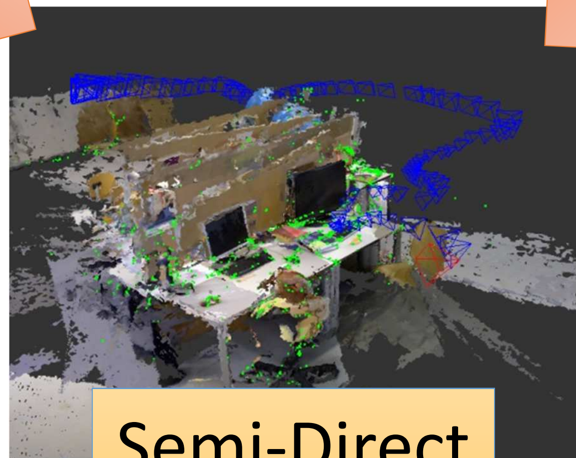
Visual SLAM – Different Ways



Feature



Direct



Semi-Direct

Visual SLAM – Feature vs. Direct

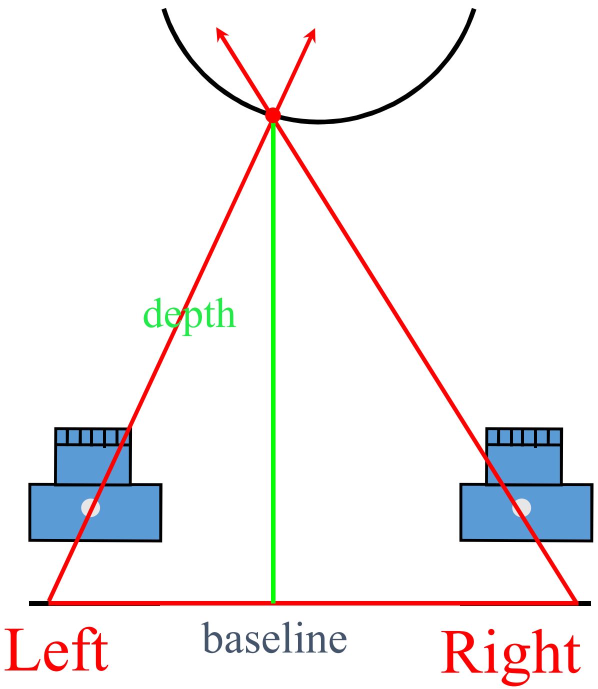
Feature-based Methods

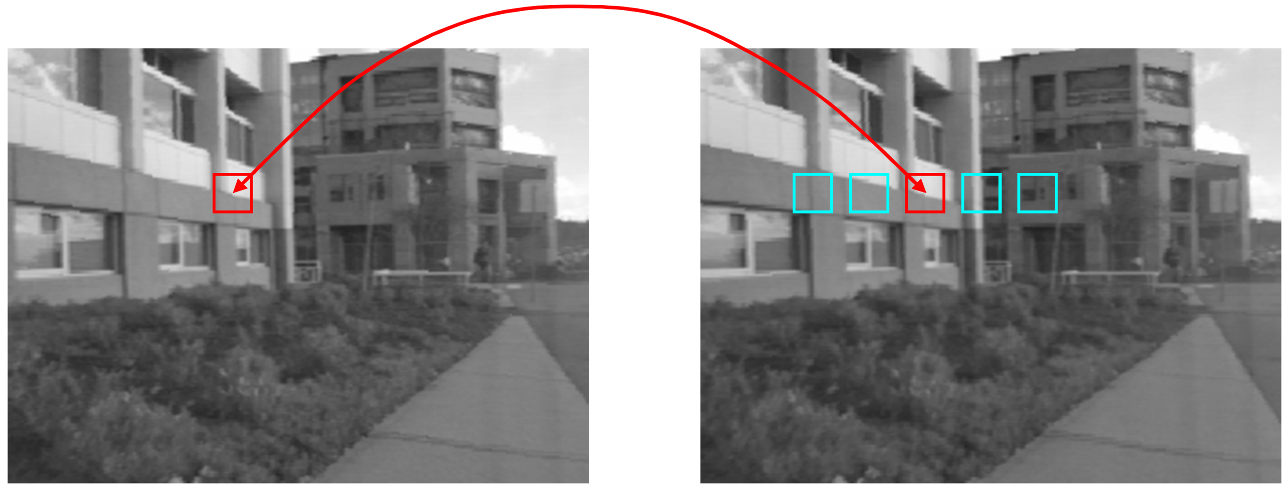
- Select 100-1000 representative points (or lines, planes) and discard the others.
- Estimation the motion from the key-points.
- Track the key-points using descriptors.
- Sparse
- Robust to outliers

Direct methods

- Estimate the motion directly from pixels.
- Use all information from images.
- Dense
- Slower
- Difficult to remove the outliers
- Needs good initialization

Visual SLAM – Basic Theory

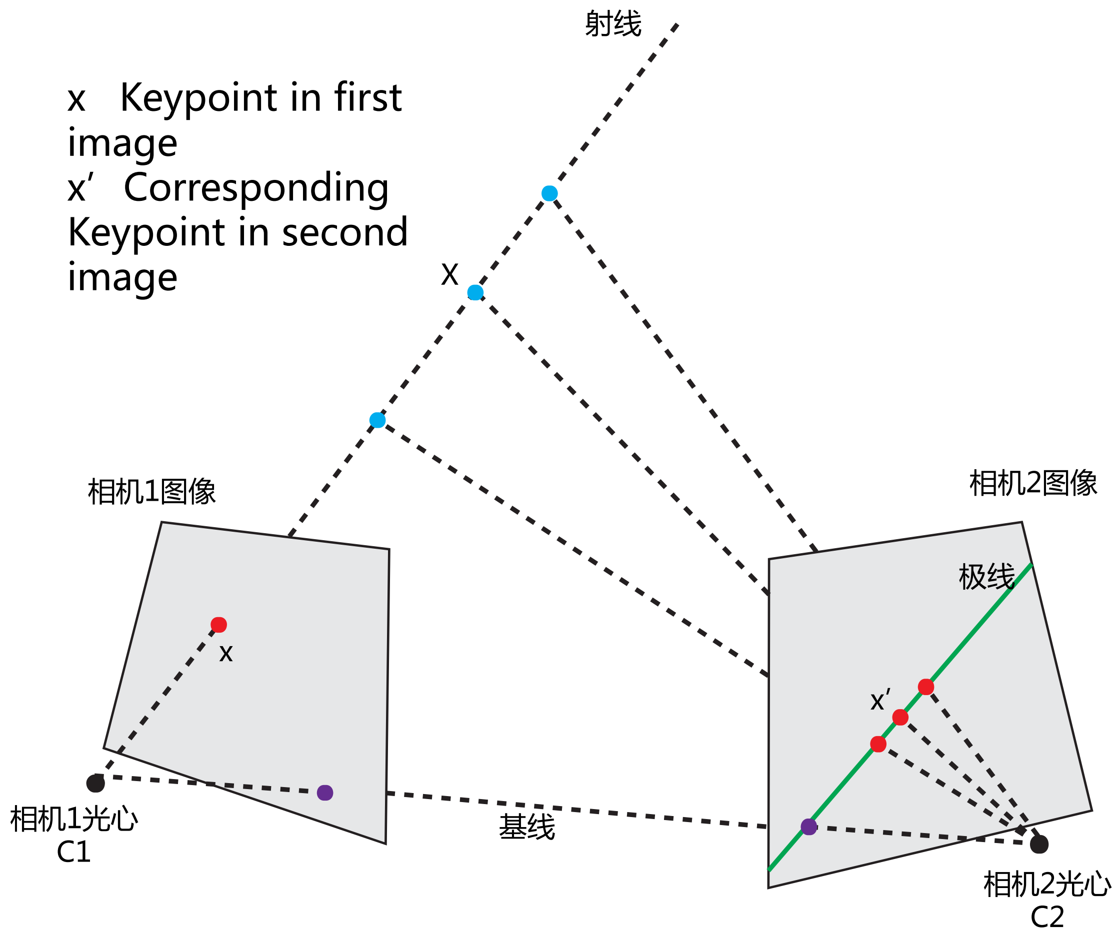




Triangulation same point in two images to get the 3D point.

Visual SLAM – How to Estimate Camera Position and Orientation

x Keypoint in first image
x′ Corresponding Keypoint in second image



$$x'^{T} F x = 0$$

Estimating F by several keypoints correspondence

↓ *F – Fundamental Matrix*

$$\mathrm{E} = \mathrm{K}^{\mathrm{T}}\mathrm{F}\mathrm{K}$$

Estimate E throuh F and K

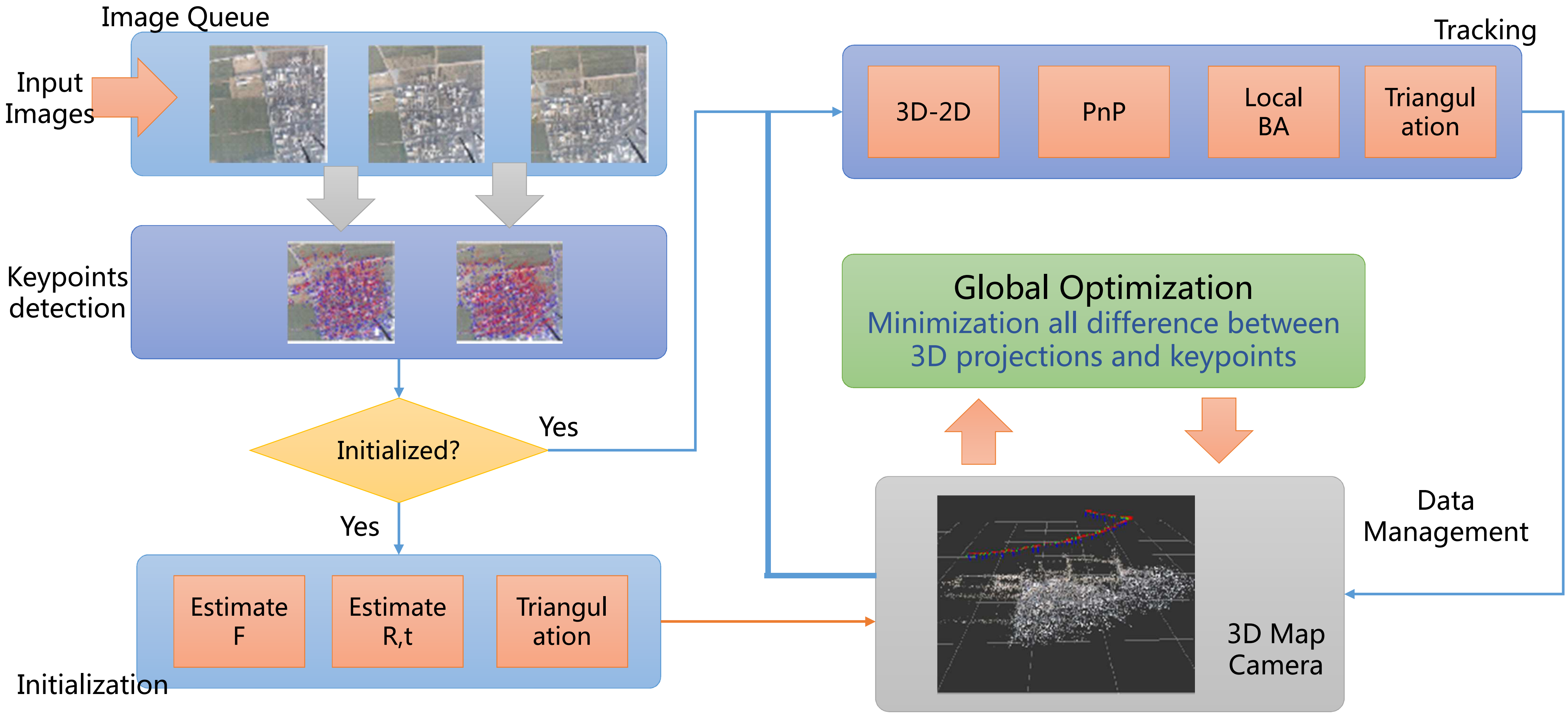
↓ *E – Essential Matrix*

R, t

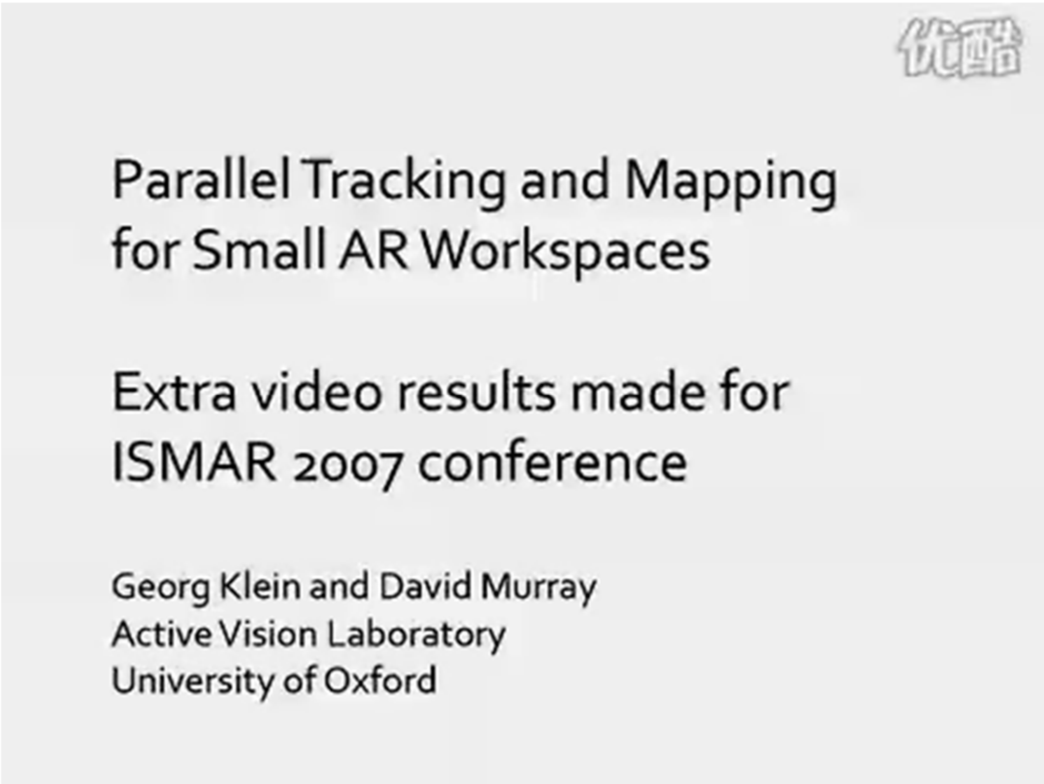
Get relative position and orientation of second image

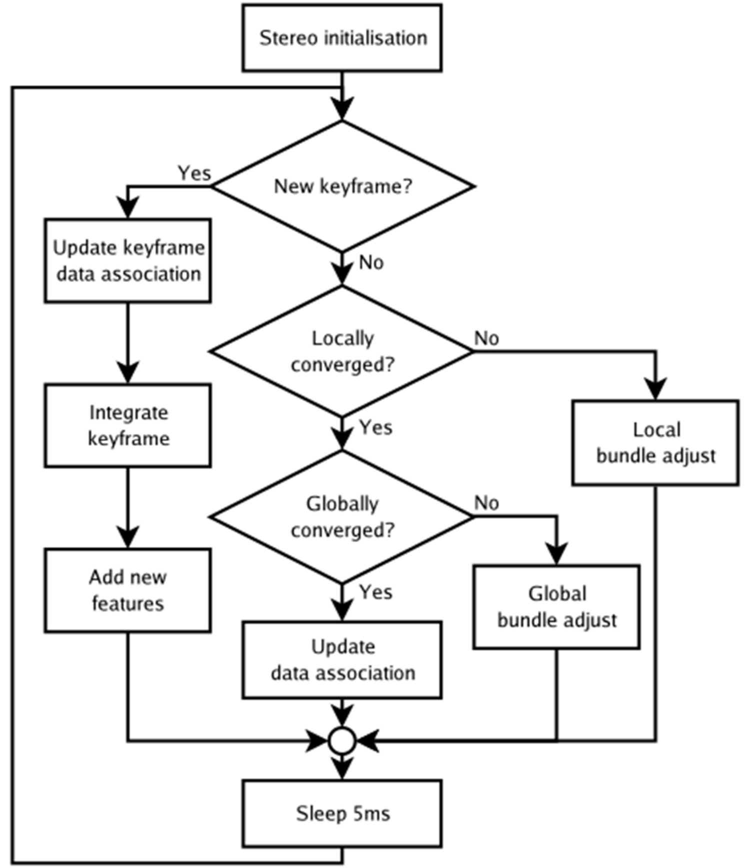
Assuming the first camera is located at origin, then the second camera’s [R|t] can be estimated.

Visual SLAM – Feature Based Methods



Feature Based Methods - Parallel Tracking and Mapping





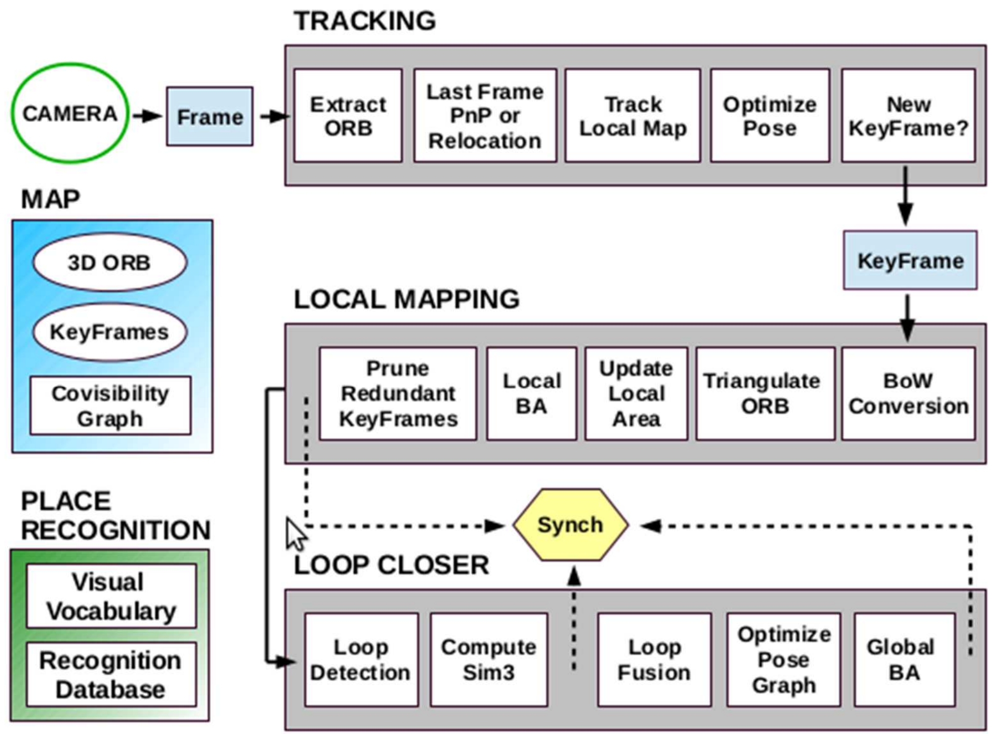
G. Klein, D. Murray, Parallel Tracking and Mapping for Small AR Workspaces, 2007

Feature Based Methods - Parallel Tracking and Mapping

- Tracking and Mapping are separated, and run in two parallel threads
- Mapping is based on keyframes, which are processed using bundle adjustment
- The map is densely intialised from a stereo pair
- New Points are initialised with an epipolar search
- Large numbers of points are mapped

G. Klein, D. Murray, Parallel Tracking and Mapping for Small AR Workspaces, 2007

Feature Based Methods – ORB-SLAM



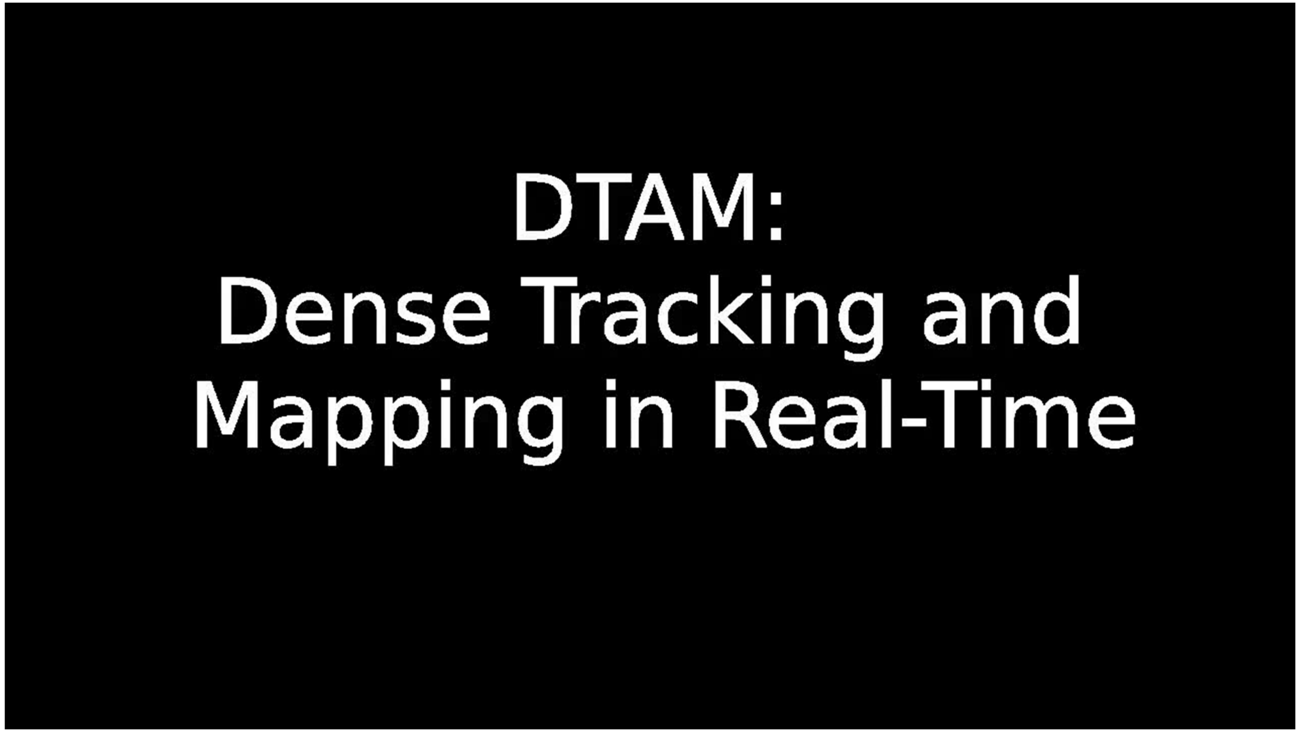
Raúl Mur-Artal, J. M. M. Montiel and Juan D. Tardós. ORB-SLAM: A Versatile and Accurate Monocular SLAM System, 2015

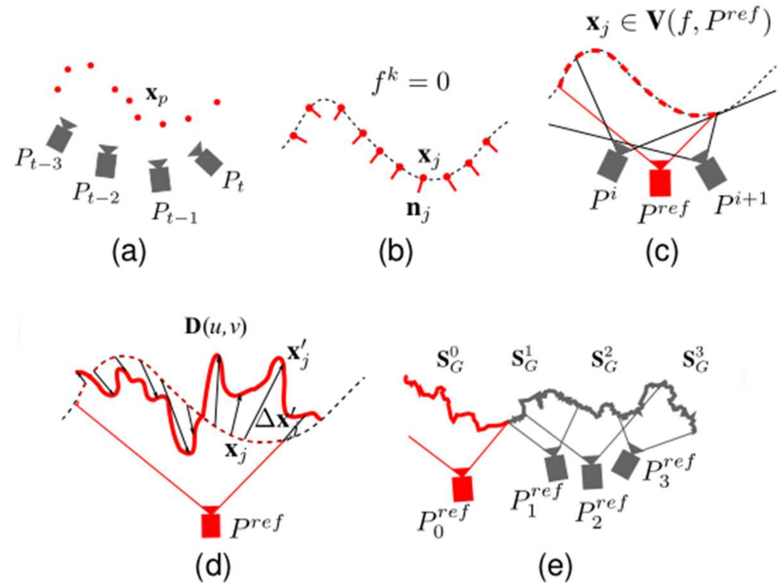
Feature Based Methods – ORB-SLAM

- Covisibility information to operate at large scale
- BoW based place recognition system for relocalisation and loop closing

Raúl Mur-Artal, J. M. M. Montiel and Juan D. Tardós. ORB-SLAM: A Versatile and Accurate Monocular SLAM System, 2015

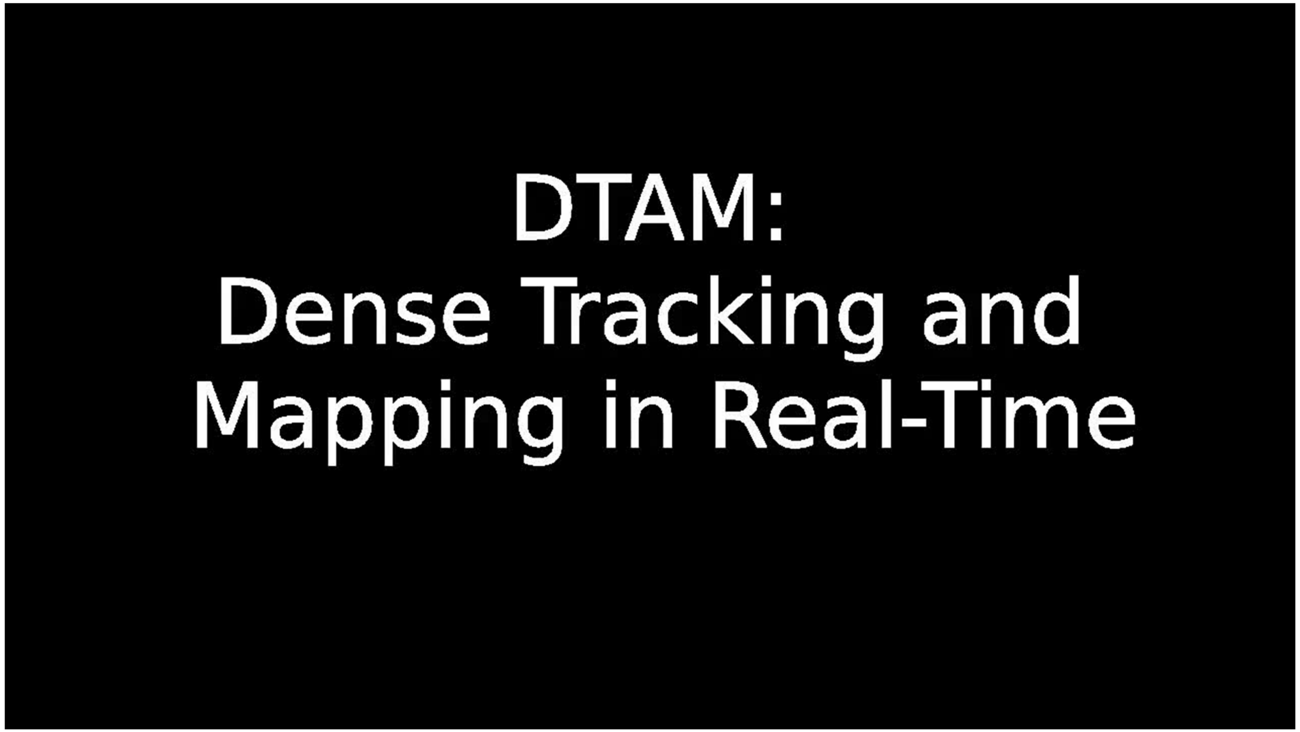
Direct Based Methods – Dense Tracking and Mapping (DTAM)





R.A. Newcombe, S.J. Lovegrove and A.J. Davison, DTAM: Dense Tracking and Mapping in Real-Time, 2011

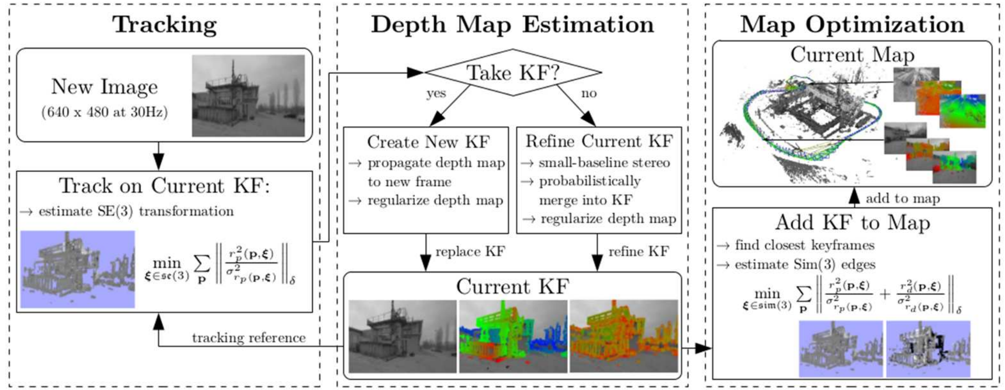
Direct Based Methods – Dense Tracking and Mapping (DTAM)



- Utilizing a coarse base surface model as the initial starting point for dense reconstruction
- Depth map creation is pipelined, and multiple depth maps are straightforwardly fused to create complete scene reconstructions
- Using GPU to accelerate the speed

R.A. Newcombe, S.J. Lovegrove and A.J. Davison, DTAM: Dense Tracking and Mapping in Real-Time, 2011

Direct Based Methods – Large-Scale Direct Monocular SLAM (LSD-SLAM)



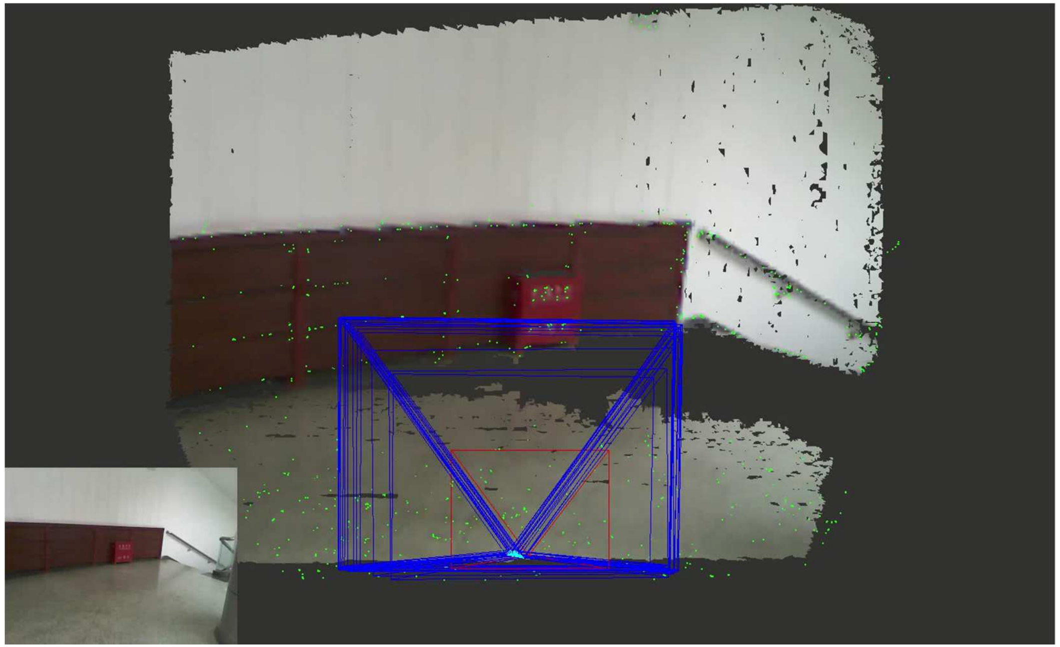
J. Engel, T. Schops, and D. Cremers, LSD-SLAM: Large-Scale Direct Monocular SLAM, 2014

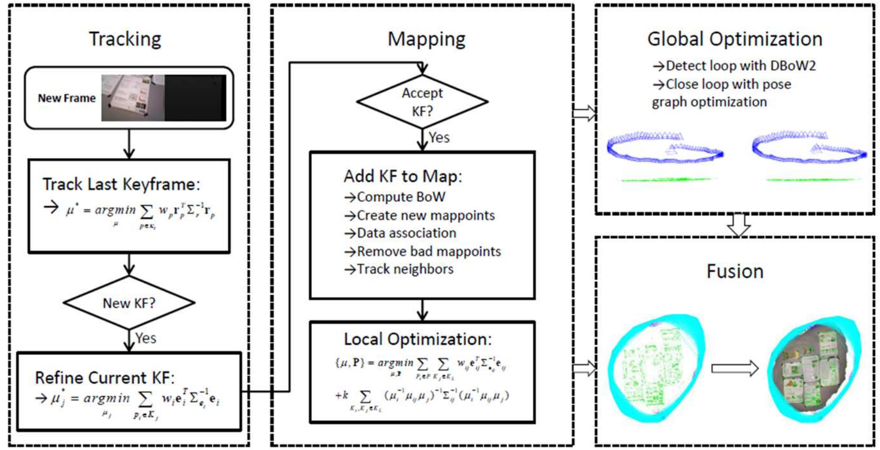
Direct Based Methods – Large-Scale Direct Monocular SLAM (LSD-SLAM)

- Using direct image alignment coupled with filtering-based estimation of semi-dense depth maps
- Probabilistically consistent incorporation of uncertainty of the estimated depth into tracking

J. Engel, T. Schops, and D. Cremers, LSD-SLAM: Large-Scale Direct Monocular SLAM, 2014

Semi-direct tracking and mapping (SDTAM)



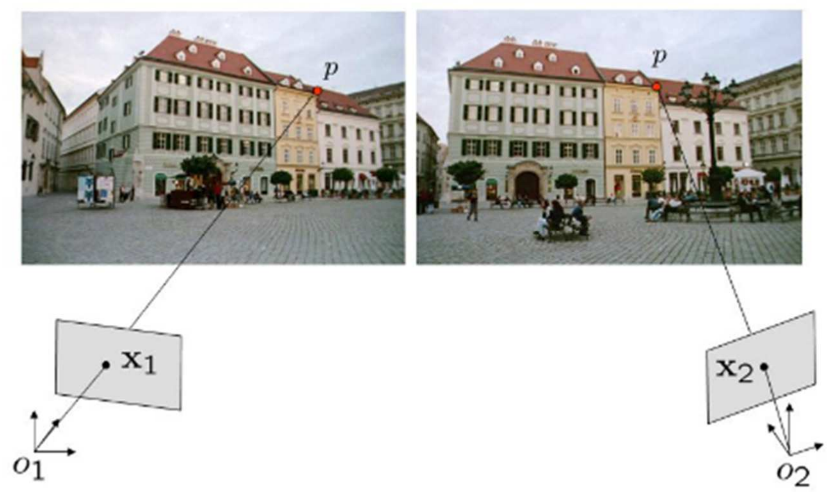


- The direct method is adopted to track current motion with high-speed, followed with a motion refinement based on feature correspondences.
- Balance between accuracy and efficiency can be realized.

Shuhui Bu, Yong Zhao, et al., Semi-direct Tracking and Mapping with RGB-D Camera for MAV, MTAP, 2016.

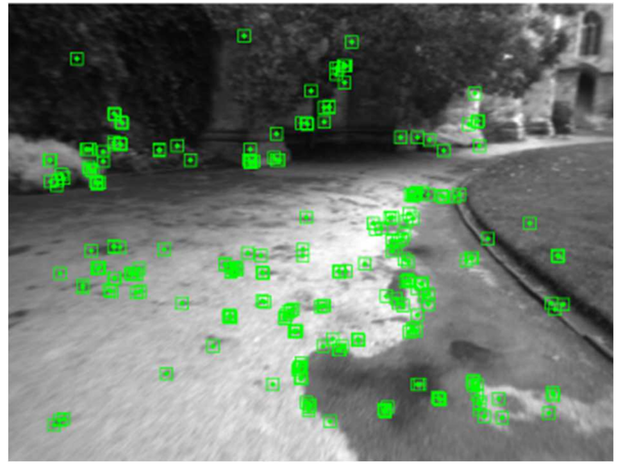
Visual Odometry - Introduction

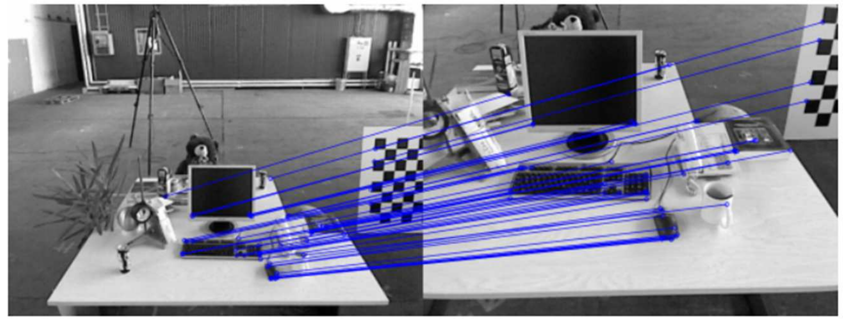
- Basic Idea
  - Estimate the ego-motion between frames.
  - Basically Two-view geometry.
- Problem
  - Data: a set of images
  - Goal: Estimate the camera motion and reconstruct the environment
  - Structure-from-Motion (SfM) or vSLAM



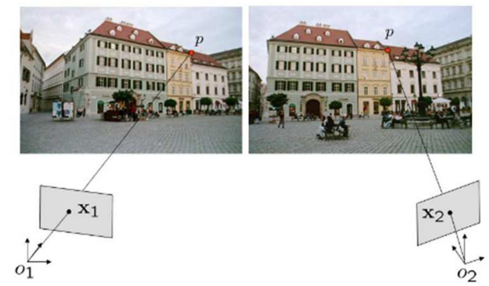
Visual Odometry - Pipeline

- **Feature-based Methods**
- Steps:
  1. Extract feature key-points and descriptors.
     - Common features: FAST, SIFT, SURF, ORB
  2. Find the corresponding matches.
     - Brute-force or kNN match.
  3. Estimate the ego-motion.
     - PnP or bundle adjustment.
- Comments:
  1. Extraction and matching cannot be always guaranteed to be successful.
  2. Tracking may lost if the motion is too fast.
  3. Ego-motion solution does not always exist or global optimal.





Visual Odometry - Pipeline



- Ego-motion estimation
- Assume a point X is observed in two frames whose pixel positions are x_1, x_2
- Pose from camera 1 to 2: R, t , Camera matrix: C

$$\boxed{\lambda_1 x_1 = CX, \lambda_2 x_2 = C(RX + t)}$$

$$X = [x, y, z]^T$$

$$x_1 = [u_1, v_1]^T, x_2 = [u_2, v_2]^T \in R^2$$

- Known: x_1, x_2, C Goal: R, t, X
- Bundle Adjustment: minimization of the re-projection error:

$$C = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boxed{\min_{X,R,t} \sum_{j=1}^{N} \|\lambda_1^j x_1^j - CX_j\|^2 + \|\lambda_2^j x_2^j - C(RX_j - t)\|^2}$$

- Unfortunately: non-linear, non-convex, on Lie manifold $\Rightarrow$ Difficult to get a global optimal solution.

Visual Odometry - Pipeline

- **Feature-based Methods**

How to solve $$\min_{X,R,t} \sum_{j=1}^{N} \|\lambda_1^j x_1^j - CX_j\|^2 + \|\lambda_2^j x_2^j - C(RX_j - t)\|^2$$

Via Epipolar Geometry:

- Build epipolar constraints:

$$\hat{x}_1^j \hat{T} R x_2^j = 0$$

- Find the Essential matrix or fundamental matrix:

$$E = \hat{T} R$$

- Solve R, t from E .

Via Optimization:

- Guess a good initial value (from last frame, constant velocity models or other sensors).
- Iterate! (Gauss-Newton or LM)

Visual Odometry – Scale?

- **Feature-based Methods**
  - Monocular SLAM: reconstruct the 3D points: X_j
  - Scale indeterminate problem:

$$\min_{X,R,t} \sum_{j=1}^{N} \|\lambda_1^j x_1^j - C X_j\|^2 + \|\lambda_2^j x_2^j - C(R X_j - t)\|^2$$

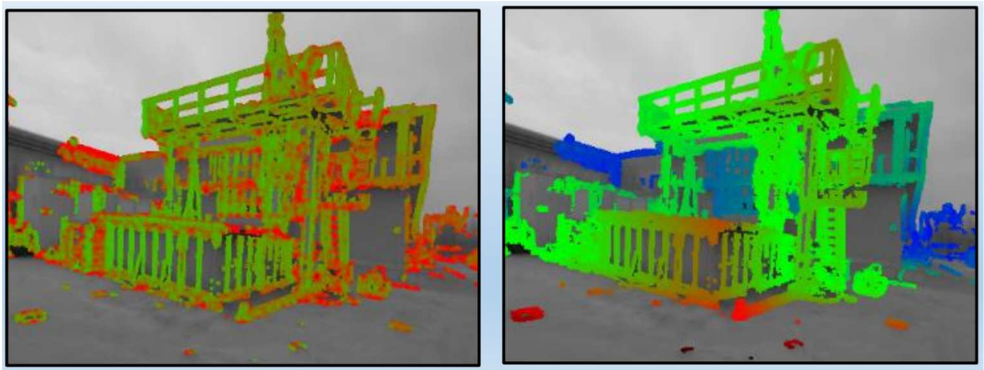
4m? 40m? 1.8 m?

- Setting relative base-line to 1 => no pure rotation

Visual SLAM – Direct Methods

- Direct Methods

$$\min E(\xi) = \sum_{x \in \Omega_{KF}} \left(I_{KF}(x) - I\big(\omega(x, D_{KF}(x), \xi)\big)\right)^2$$



I: Image gray scale value
D: Depth map of pixels
ω: Projection function

- Minimization the gray scale values of pixels.
- Assume the camera moves slowly, smoothly and the light condition does not change much.
- Reconstruct dense results instead of sparse feature points.

Visual SLAM – Summary

- So in visual odometry, we
  - Track feature points (lines, planes);
  - Estimate the ego-motion between consecutive frames;
  - Reconstruct the local map (sparse key-points or dense models);

- But they may:
  - Drift during the motion;
  - Inconsistent with other parts of environments;
  - Lost due to occlusion or fast motion.

Solutions

- Global optimization
- Loop closure
- Re-localization

Optimization

Optimization

- In history, we have two kinds of back-ends
  - Filter
  - Optimization
- In filter methods, we use past information to maintain the current status.

$$x = \left[x_b, x_{L1}, \ldots, x_{Ln} \right]$$

Status Variable

$$x_k^- = f(x_{k-1}, u_{k-1})$$
$$P_k = AP_{k-1}A^T + WQW^T$$

Predict

- Extended Kalman Filter

Motion: $x_p^{i+1} = f(x_p^i, u_i) + w_i$

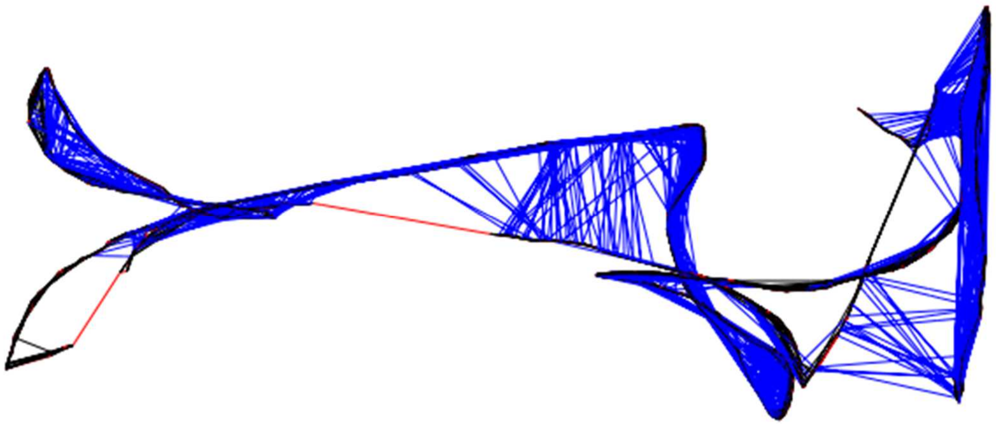
Observation: $z_{i,j} = h(x_p^i, x_L^j) + v_{i,j}$

$$h = h\left(x_k^-, 0\right)$$
$$K = PH^T \left(HPH^{-1} + VRU^T \right)^{-1}$$
$$x_k^+ = x_k^{-1} + K(z - h)$$

Update

Global Optimization

- Global optimization: reduce the drifts



Full-SLAM or Graph-based SLAM

- Considering all the past observations and put them into a large optimization problem.
- Usually represented as a Graph:

$$\boxed{G = \{V, E\}}$$

- Vertex: the optimization variables
- Edges: the error terms (or constraints)
- A large MSE:

$$\boxed{\min \sum_{k=1}^{N} e_k(z_k, x_k)^T \Omega_k e_k(z_k, x_k)}$$

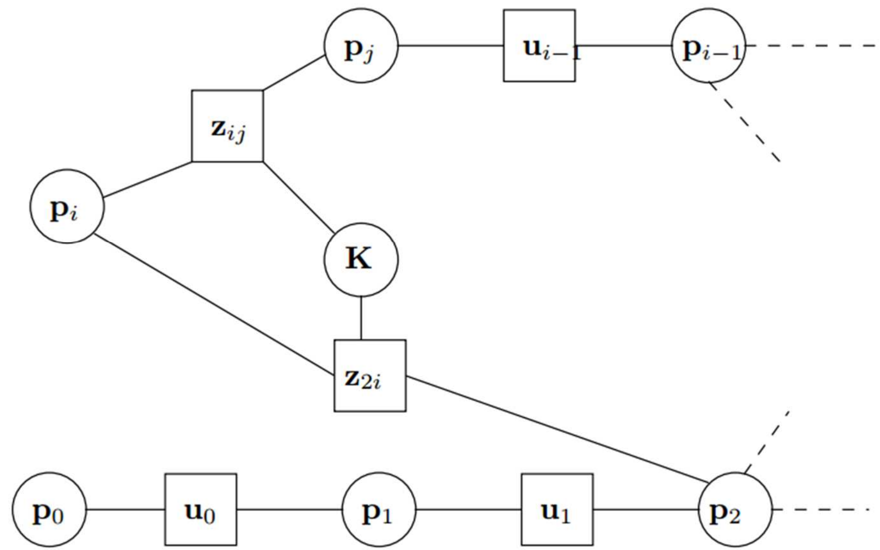
Global Optimization – Graph and Sparse

- Put them together to build a error function
- Guess a good initial value and then iterate it!
- Just a large bundle adjustment.

- Advantages:
  - **Use more information than filters.**
  - Convenient to represent the loops.
  - **Sparse structure in the graph helps computation!**
  - **Mix different vertices and edges!**

- Disadvantage:
  - Hard to maintain the graph size.
  - Global optimization still needs more computation time comparing with filters.

- Tools: g2o, ceres, etc.



A sample graph

P: poses of the camera (or robot)
Z: observations
K: uncalibrated parameters
U: motion constraints

Global Optimization – Loop Constraints

The loop constraints can help to correct the drift.

Global Optimization – Summary

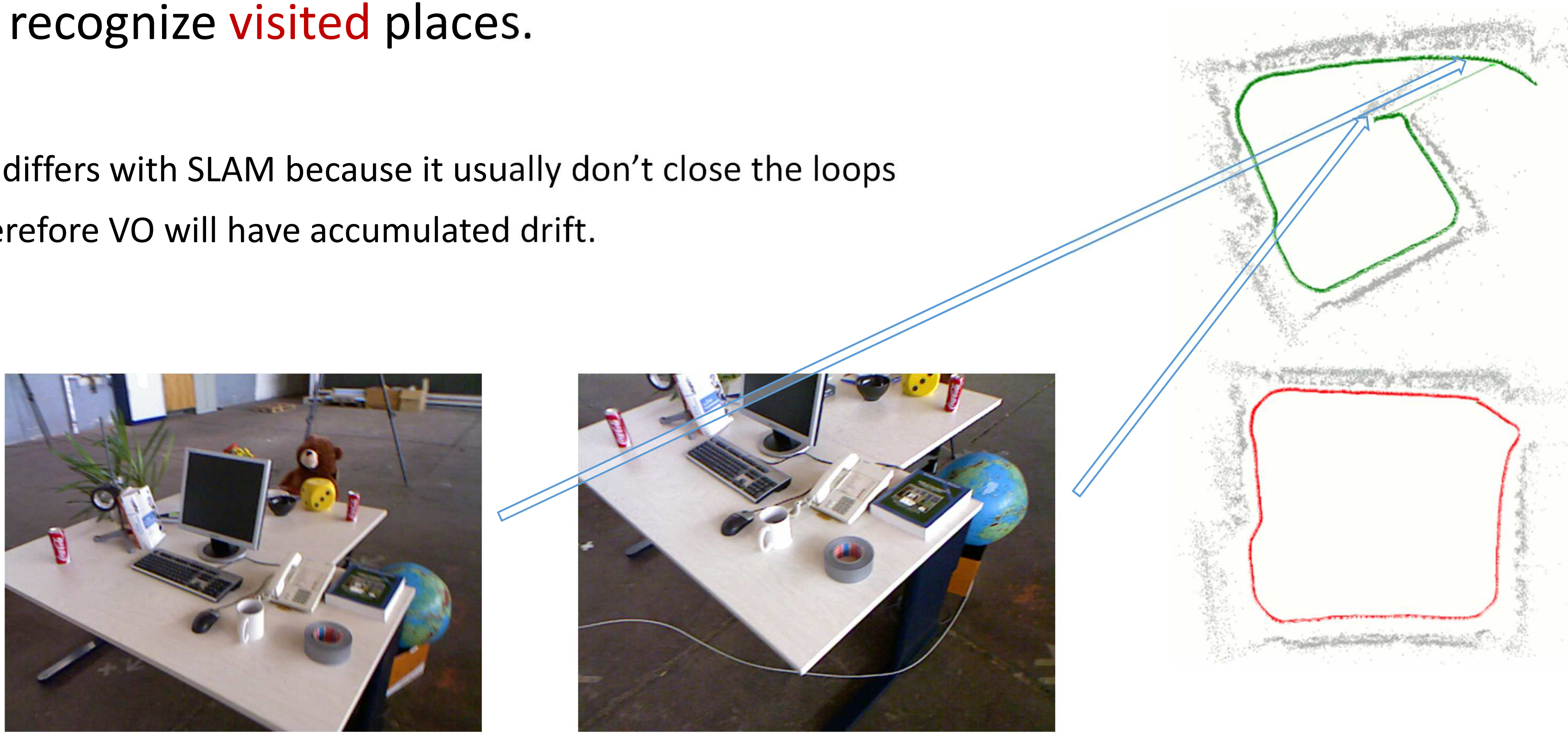
- Problems in graph optimization:
  - What happens if I add wrong edges into the graph?
    - Robust error term
  - What if VO gets lost?
    - Measurement of the connectivity
  - The graph will grow over time.
    - Long-term SLAM, needs pruning
  - How to fuse two different graph?
    - Multi-robot SLAM

Loop Closure

Loop Closure

- What is loop closure?
- To recognize visited places.

- VO differs with SLAM because it usually don’t close the loops
- Therefore VO will have accumulated drift.



Loop Closure

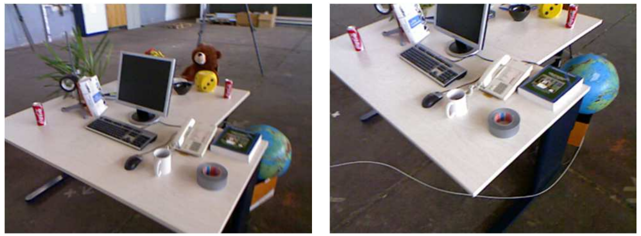
- The Key of closing loops in vSLAM:
  - Are the positions of cameras near to each other?
  - Are the images look the same?

Odometry-based Approaches

- Assume the estimated posts are accurate enough.
- Recursive in logic.

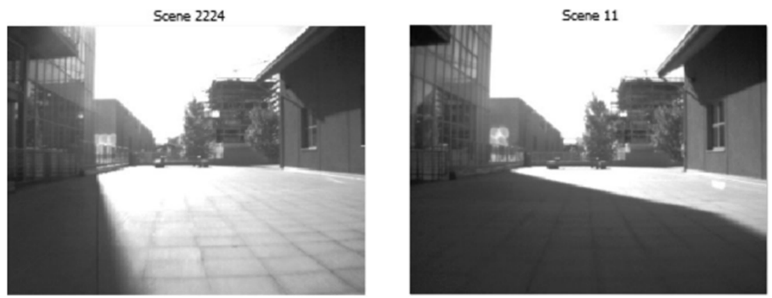
Appearance-based Approaches

- Only consider the observed image.
- State of the art in vSLAM.



YES

Looks same but not



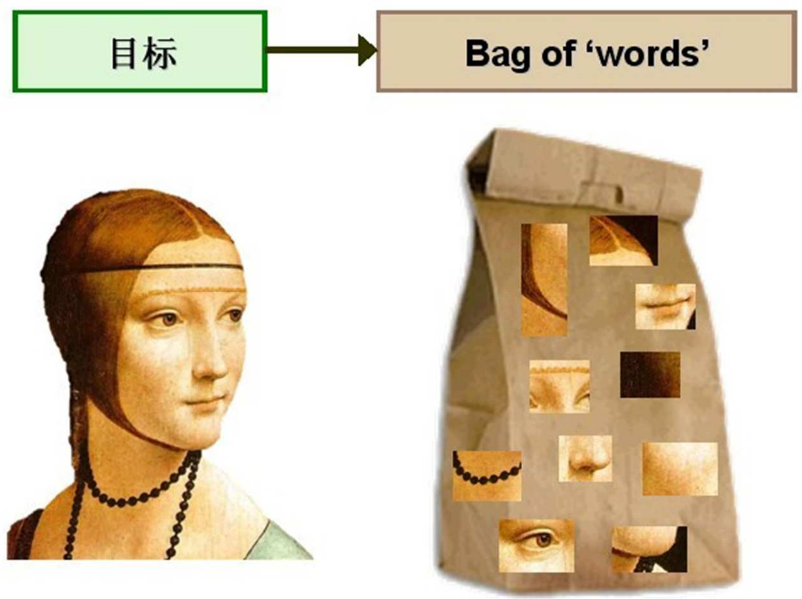
Looks different but the same

False Positive

False Negative

Loop Closure

- How to measure the similarity of images?
  - Directly A-B ? -What if the view and light changes?
- We need more complicated models: **Bag-of-Words**



From many images: nose, eyes, hairs, …

↓

Dictionary

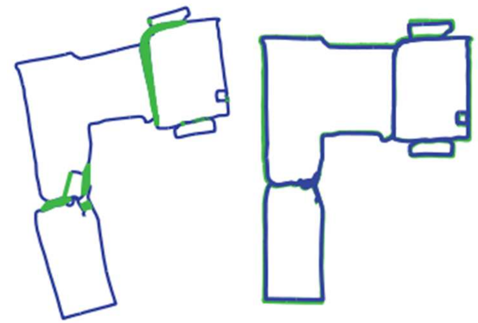
↓

Face = 1 nose + 2 eyes + 1 hairs + …

↓

Features -> Words

Loop Closure



- Bag-of-Words loop closure approaches:

A large image set → Features: SIFT, ORB → Clustering → Dictionary

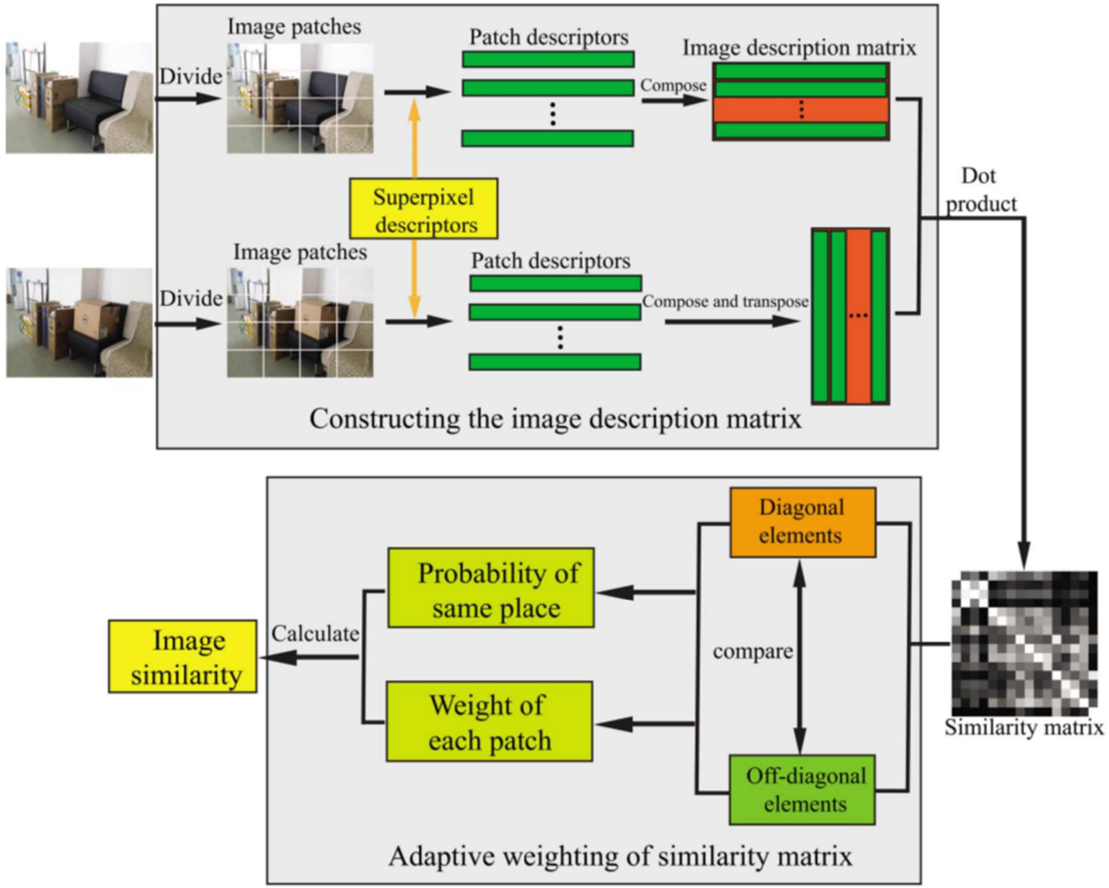
Image A → Dictionary → Words A= 2 car + 1 people

Image B → Dictionary → Words A= 1 car + 1 cat

⇒ Compute Similarity and Raise loop hypothesis

- How to evaluate loop closure methods?
  - Precision-recall curve
- Approaches from ML: Auto-encoders, CNNs, etc.

Loop Closure – Deep Learning



- Using CNN to extract representative feature
- Considering spatial relationship between objects in scene

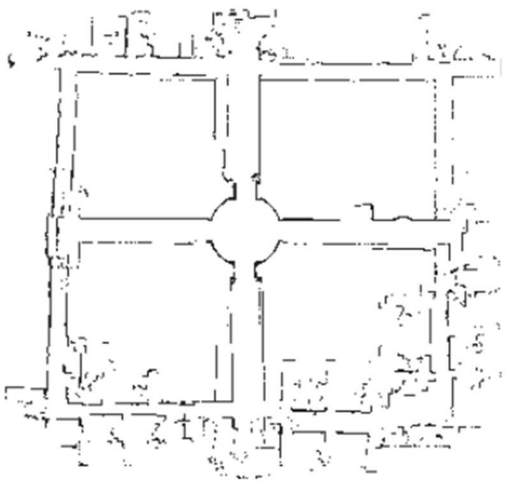
Qing Li, et al., Place Recognition Based on Deep Feature and Adaptive Weighting of Similarity Matrix, Neurocomputing, 2016.

Part of source codes: http://www.adv-ci.com/blog/source/pi-cnn and http://www.adv-ci.com/blog/source/pi-slic

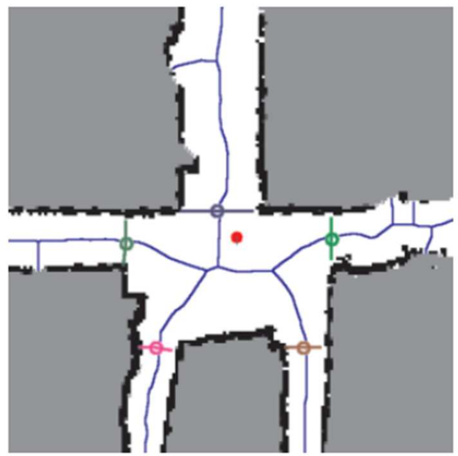
Map

Map

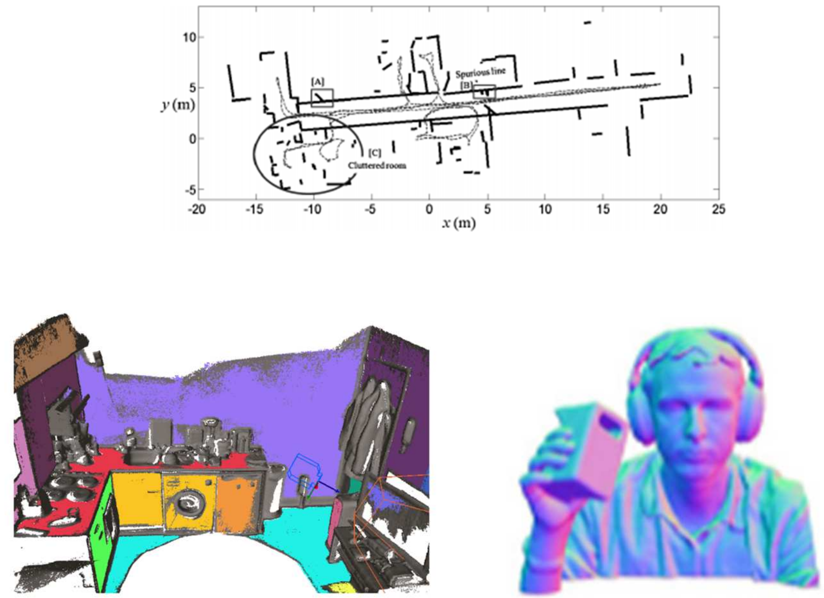
- Map is one of the major outputs of SLAM
  - However it depends on what is SLAM applied for.



Metric map



Topological map



Others

Map

In mapping we seek an efficient way to represent the environment.

Metric Maps

- Accurate
- Navigation

- Hard to extend
- Expensive in storage

- Examples:
  - Grid map
  - Occupancy map
  - Raw point cloud map

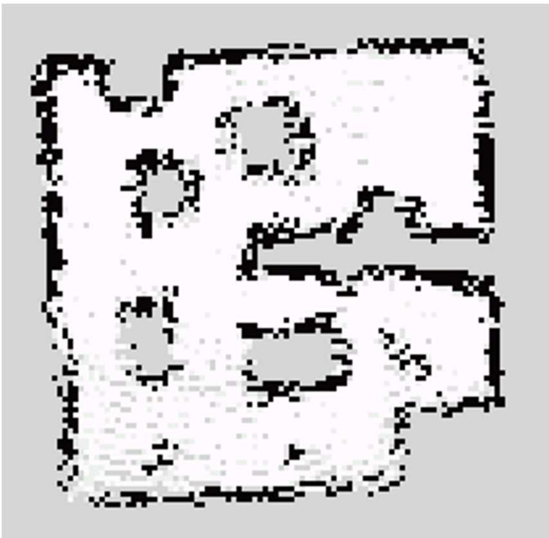
Topological Maps

- Flexible
- Low-cost

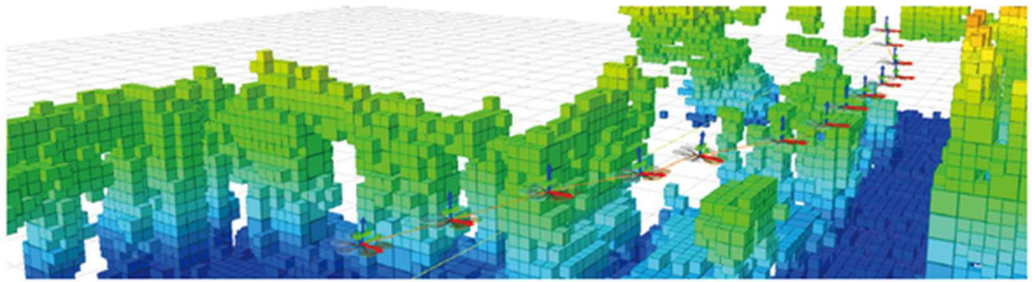
- Inaccurate
- Need local models to navigate

Map

- Maps used in Navigation:
  - Occupancy maps: model the possibility if a point is occupied



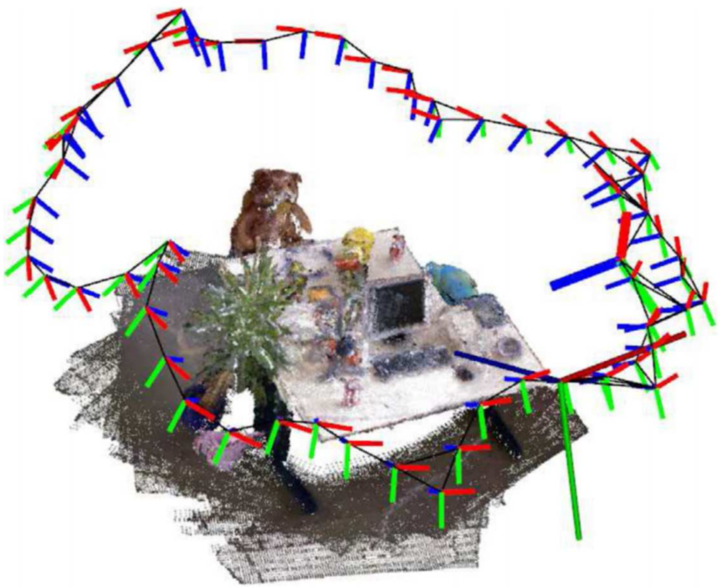
2D Occupancy map (ROS)

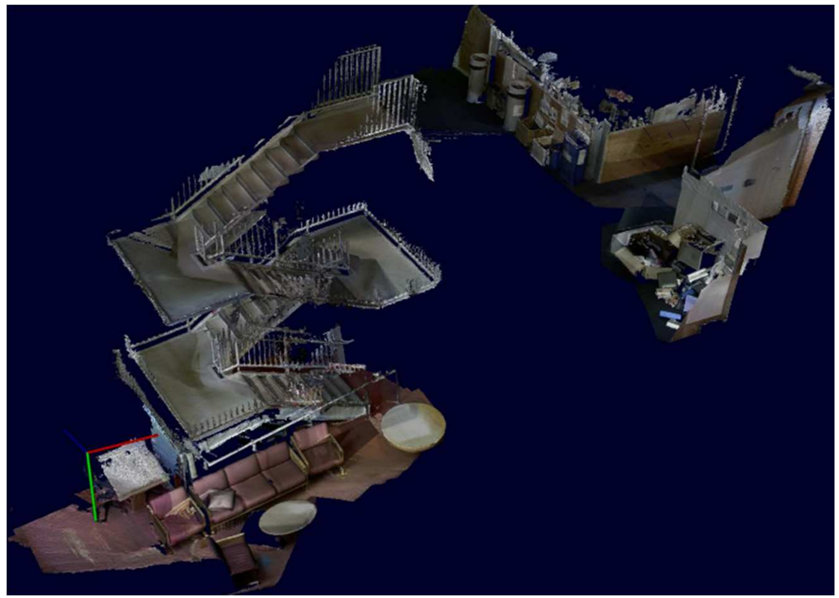


3D Occupancy map (Octomap)

Map

- Maps used in Reconstruction:
  - TSDF (Truncated signed distance function)
  - Surfels



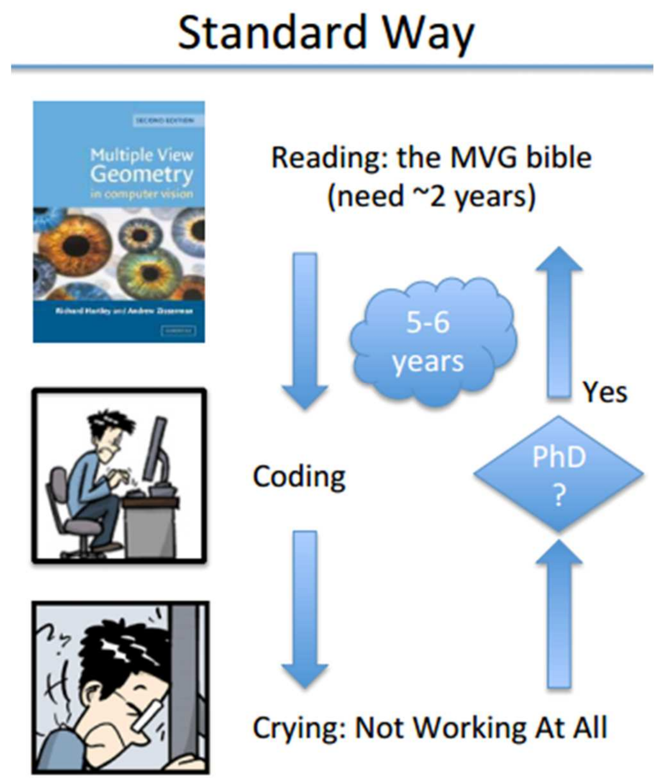


Conclusion

Conclusion

- In a typical SLAM system, we use
  - VO to estimate the ego-motion between frames
  - Optimization to handle the global trajectory
  - Loop closure to correct the draft
  - Map to describe the environment

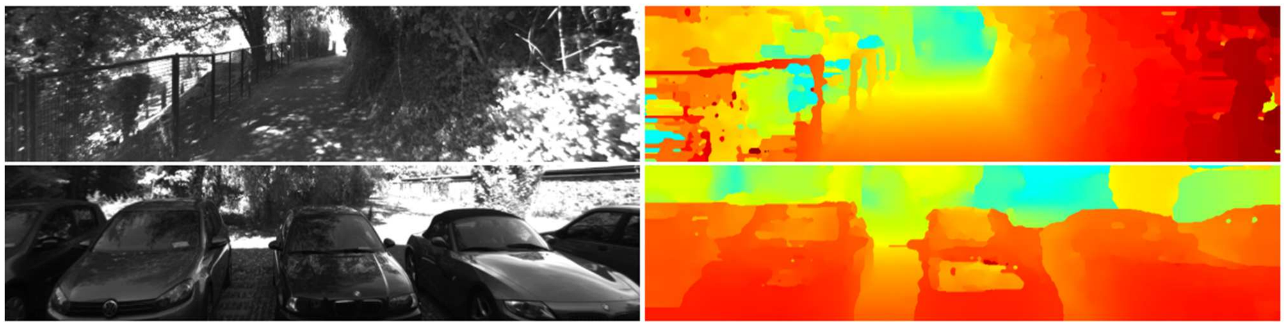
- Is that all about SLAM?
  - We haven't talked about coding yet.



LSD-SLAM Analysis: *http://blog.csdn.net/lancelot_vim*

Future of SLAM

- Semantics are necessary to build bigger and better SLAM systems.
- Will end-to-end learning soon replace the mostly manual labor involved in building today's SLAM systems?
- Use SLAM to fuel Deep Learning



Open-source Codes

Qualified open-source SLAM solutions

Name	Site
Rgbd-slam-v2	https://github.com/felixendres/rgbdslam_v2
ORB-slam	https://github.com/raulmur/ORB_SLAM
LSD-slam	https://github.com/tum-vision/lsd_slam
Hector-slam	https://github.com/tu-darmstadt-ros-pkg/hector_slam
SVO	https://github.com/uzh-rpg/rpg_svo
RTABslam	https://github.com/introlab/rtabmap_ros#rtabmap_ros
Dvo-slam	https://github.com/tum-vision/dvo_slam
Kinect Fusion	http://research.microsoft.com/en-us/projects/surfacerecon/
Kinfu_large_scale	http://pointclouds.org/documentation/tutorials/using_kinfu_large_scale.php
DTAM	https://github.com/anuranbaka/OpenDTAM

Conclusion

- For a student starting to work in SLAM, one should learn:
  - Math: geometry, probabilistic even ML
  - Coding skills:
    - Linux
    - C++
    - Libraries: ROS, OpenCV, PCL, g2o, DBoW2, libpointmatcher, octomap, Fabmap, ceres …
    - Python
- Future issues:
  - Dynamic, Multi-robots, Semantic, Low-weight devices, Mobile platforms