SLAM
(Simultaneous Localization and Mapping)

Index

- What is SLAM?
- Kind of SLAM
- Omnidirectional Camera
- RGB-D
- Deep Learning

1. What is SLAM

- "the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it." (Wikipedia)
- "주변 환경에 대해 로봇의 위치와 지도를 동시에 계산하는 것." (Uni-Fre)

- Probabilistic fashion

  Given:
  - Robot control signal
  - A set of observations

  Estimate:
  - Map of Landmarks
  - Robot pose

  Sources of Error
  - Control signal
  - Mapping sensor
  - Motion model
  - Observation model

'SLAM Techniques and Algorithms', Jack Collier, 2010

2. Kind of SLAM

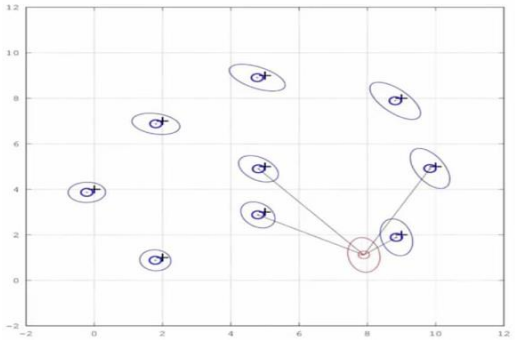
2. Kind of SLAM

- **Bayesian Filtering based**

: Prediction step(control input)과 Correction step(observation)을
반복적으로 수행하여 오차를 제거

Ex-1) EKF-SLAM

- 기존의 Kalman filter는 선형 시스템에서만 적용이 가능하므로 Taylor 선형 근사를 통해 비선형 시스템에서도 적용이 가능하게 만든 모델
- 다른 센서들의 결합을 통해 불확실성을 줄여줌
- 지도가 커질 수록 상태 벡터가 제곱을 커짐 -> Submap 방식 도입

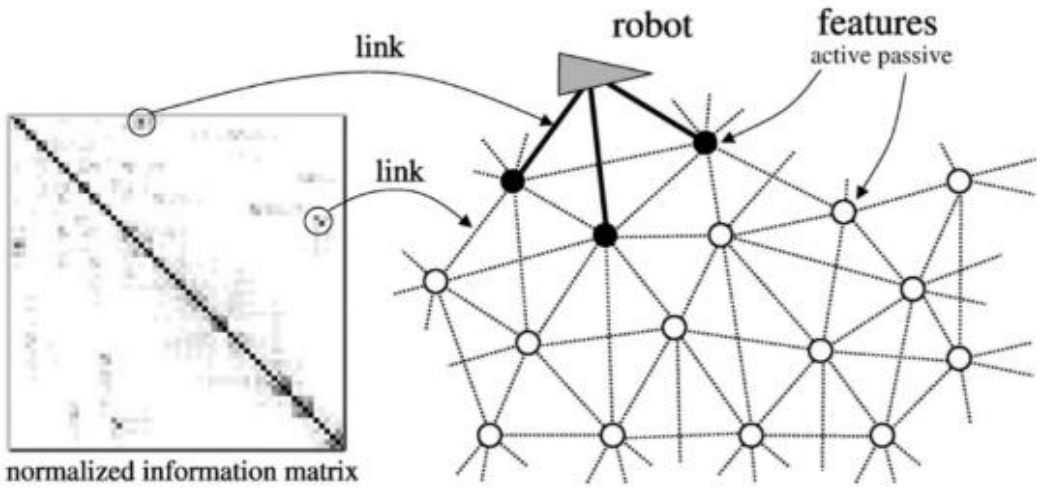


Ex-2) UKF-SLAM

- Jacobian 계산을 피하기 위해 만들어진 모델
- Sigma point들을 sampling하여 새로운 추정치를 생성
- 계산 비용(속도)가 EKF에 비하여 느림

Ex-3) IF-SLAM

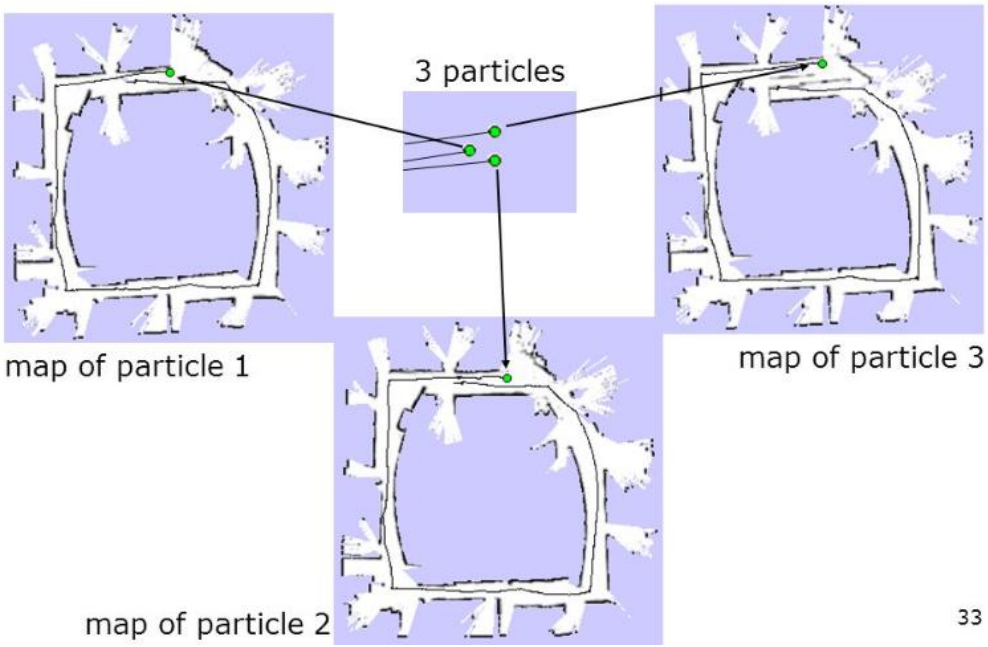
- Information matrix(covariance matrix)의 inverse matrix를 이용한 모델



- **Particle Filtering based**

: 현재 상태를 확률 밀도에 따라 입자로 표현하여 변위 예측 후 업데이트
환경 데이터 값에 따라 입자의 가중치가 바뀜

- Land mark 당 입자 집합이 필요, 이전 작업이 큰 영향력을 가짐
- Gaussian noise 가정을 두지 않음



- **Visual Odometry**

: 카메라를 이용하여 연속된 영상만으로 카메라의 혹은 카메라가 장착된 플랫폼의 자세와 위치를 추정하는 방법

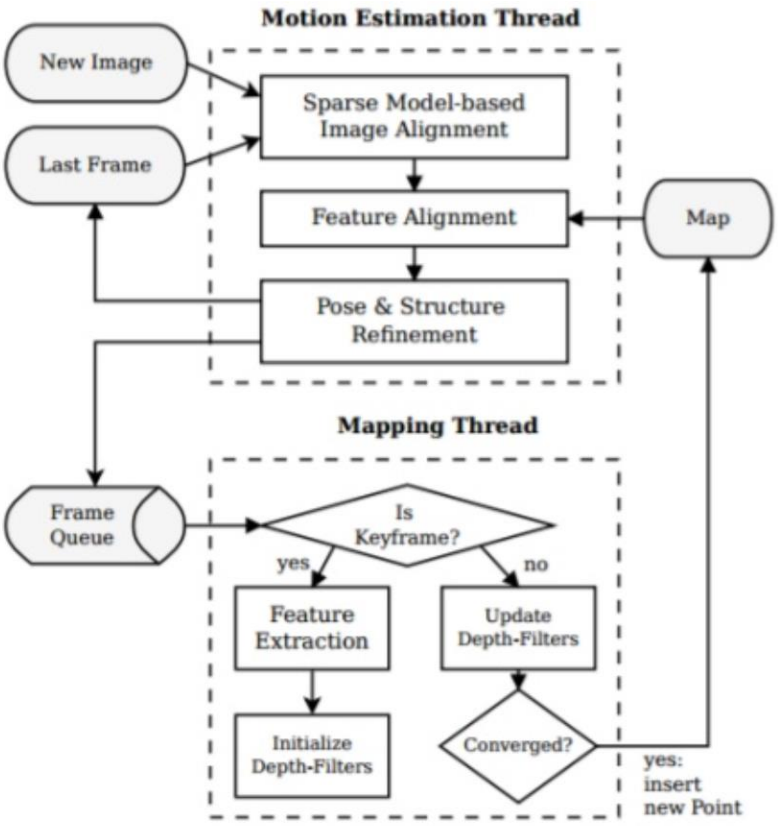
1) Cost function에 따라

Ex-1) Indirect
- 영상에서 특징 점들을 추출하여 Reprojection error를 최소화하는 방법
- 직접에 비해 계산량이 적음
- 영상의 특징 점이 필요, Motion blur에 강인하지 않음

Ex-2) Direct
- 영상의 밝기 정보를 이용하여 밝기 차를 최소화하는 방법
- 영상 내 밝기가 급격하게 변화하는 환경에서는 성능 저하

Ex-3) SVO(Semi direct Visual Odometry)

: 특징 점 기반 방법과 직접적 밝기 기반 영상의 단점을 최소화 시키기 위해 두 가지 방법을 혼합하여 만들어낸 Hybrid 모델



2) Camera 개수에 따라

Ex-1) Single Camera

- Baseline이 없으므로 각 특징 점들의 3차원 위치 추정이 어려움
- 일정 거리만큼 좌우 또는 상하로 움직인 후 나타나는 시차를 이용하여 선형성 보장, EKF 필터 적용 가능
- 스케일모호성을 가짐, scale drift가 누적될 수 있음

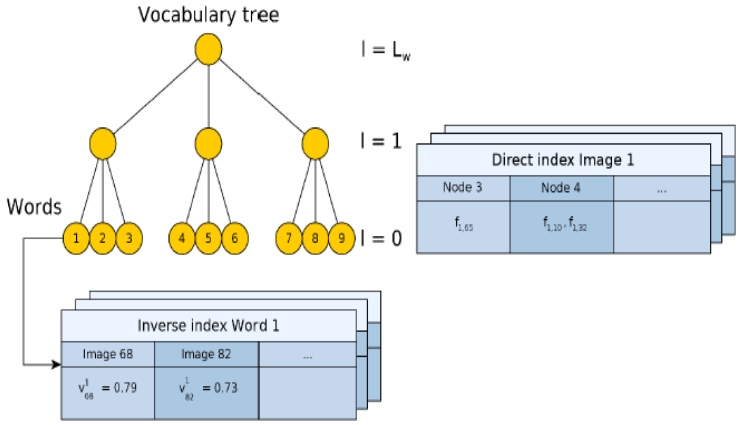
Ex-2) Stereo Camera

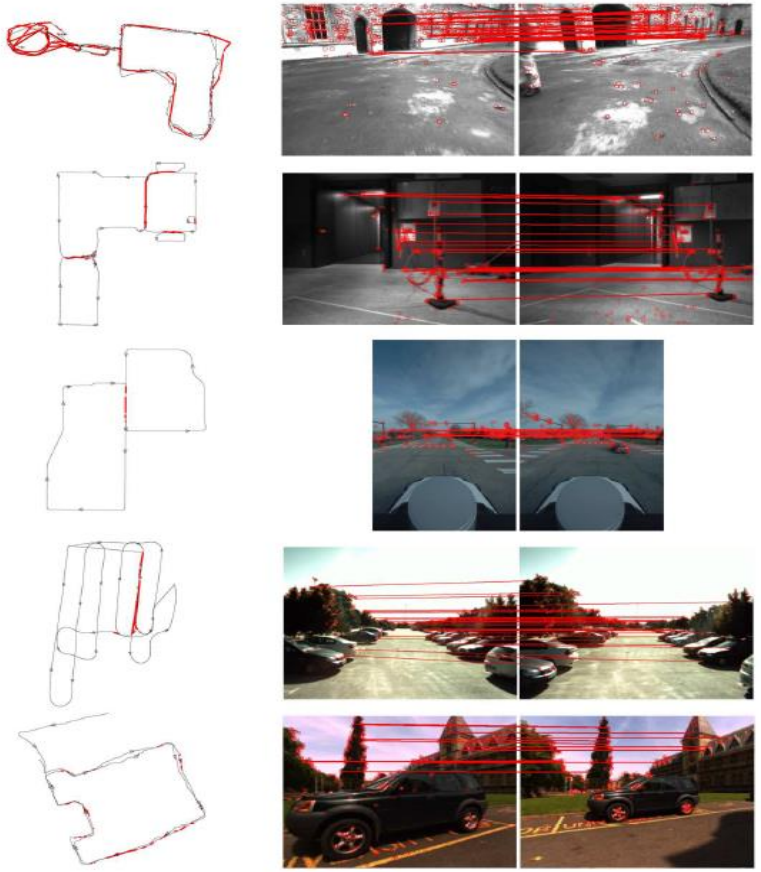
- 두 카메라 사이의 거리 벡터가 고정되어 있어 매 프레임마다 각 물체까지의 거리 측정 가능
- 카메라와 물체 사이 거리가 Baseline 보다 긴 경우에는 충분한 시차를 확보되지 않아 카메라와 물체 사이의 거리를 추정하기 어려움

- **Matching**

Bags of Binary Words for Fast Place Recognition in Image Sequences, Dorian G´alvez-L´opez and Juan D. Tard´os, IEEE Transactions on Robotics, 2012

- FAST feature + BRIEF descriptor
- Bag of words (+ direct index)
- fast performance
- Geometrical consistency

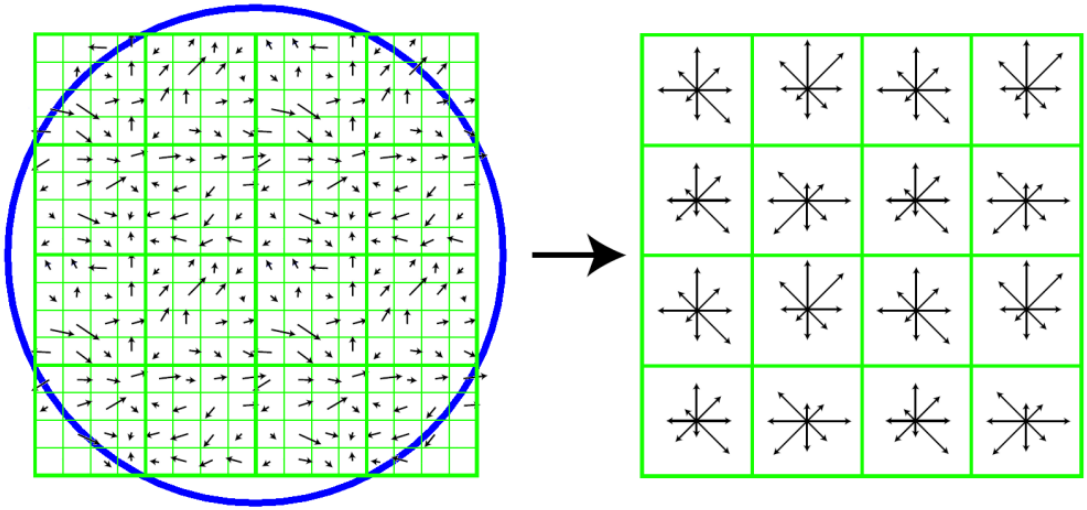




Descriptor

특징 점의 주변 특성을 이용하여 해당 특징 점을 표현하는 벡터를 만들어
이미지에서 같은 특징 점을 매칭하거나 추출할 때 사용

- 특징 점을 중심으로 16x16 영역을 4x4 크기의 16개 윈도우로 나눔.
- 윈도우의 16개 포인트에서 Gradient 벡터의 크기와 방향을 계산
- Gradient 벡터의 방향을 8개의 각도로 rounding
- 8개의 각도에 대해 Gradient 벡터의 크기를 더하여 일종의 Gradient 히스토그램 생성
- 윈도우 16개의 히스토그램을 모두 모아 특징 점 주변에 대한 정보를 128(8x16)차원의 벡터로 표현



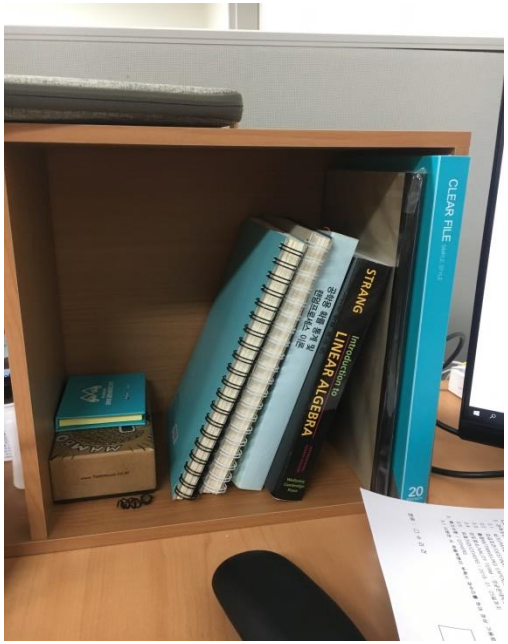
* Key point

- 물체의 Scale, Size, Orientation이 변해도 식별 가능한 점
- 물체를 바라보는 시점이나 조명이 변해도 변하지 않는 고유한 점
- ex) 다각형 꼭지점, 선분의 끝 점.

SIFT(Scale-Invariant Feature Transform)

특징 점의 크기와 각도까지 같이 계산하여 이미지의 크기가 변하거나 회전해도
동일한 특징 점을 찾을 수 있도록 하는 방법.
특징 점 근처의 이미지 특성(Histogram)도 같이 계산해서 특징 점 이미지 모양도 구별 가능.

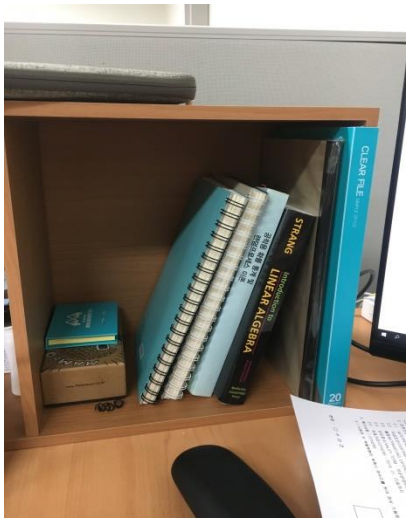
- 크기에 불변한 특징 점을 추출하기 위해 Scale-Pyramid 생성
- 각 이미지에서 특징 점을 추출, 이러한 특징 점들은 Scale-Invariant이지만 회전에는 취약
- 회전 불변 특성을 위해 특징 점 주변에 Gradient 방향과 크기를 수집
- 360도를 36등분하여 36개의 bin를 가진 Gradient vector histogram생성
- 가장 큰 bin이 해당 특징 점의 방향, 특징 점의 크기로 설정



FAST(Feature from Accelerated Segment Test)

극도의 빠른 속도를 추구하는 특징 점 추출 방법.
노이즈에 강인하지 못한 특성을 가짐.

- 픽셀 P를 중심으로 16개의 Pixel들(3픽셀의 반지름)이 원으로 둘러 쌓여져 있음
- P보다 Threshold 값 t 만큼 이상 더 밝은 픽셀들이 n개 이상 연속되어 있거나, 더 어두운 픽셀들이 n개 이상 연속되어 있는지 확인
- 만약 위에 해당한다면 P를 Corner으로 판단(Decision Tree를 이용하여 판단)

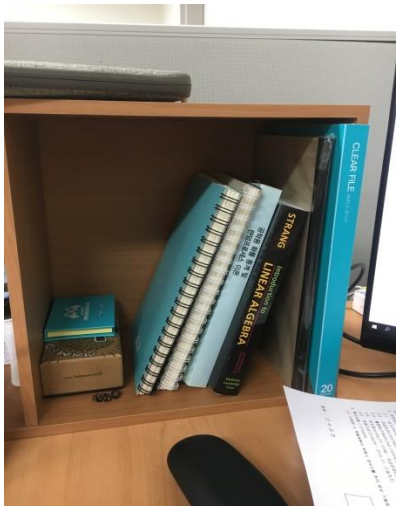


* Decision Tree
P보다 큰, P보다 작은, P와 유사한 밝기로 나누어 16차원의 벡터로 표현하고 모든 픽셀, 이미지에 대한 벡터를 쌓아 훈련
코너 점 주변 픽셀을 코너로 인식하는 것을 방지하기 위해 Non-maximal suppression 후 처리 작업

ORB(Oriented FAST and Rotated BRIEF)

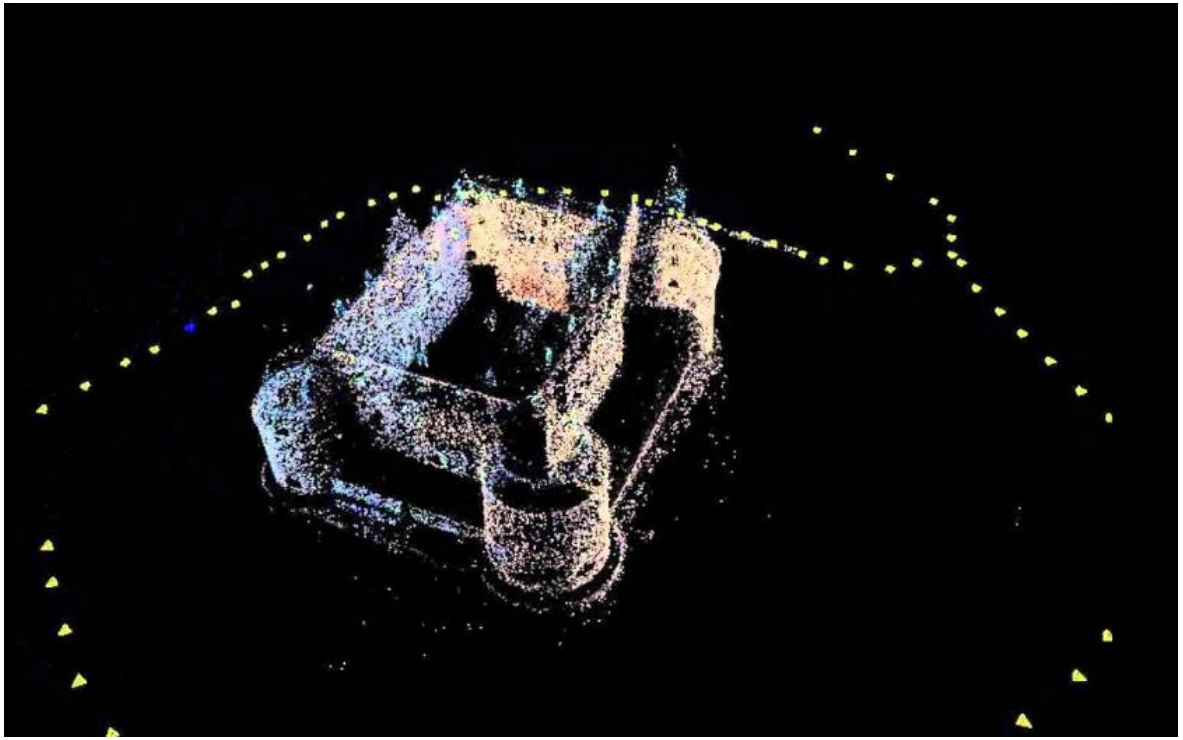
SIFT에서 하나의 특징 점을 설명할 때 128차원의 실수 벡터를 나타내므로 꽤 많은 리소스를 낭비할 수 있음.
BRIEF를 이용하여 Descriptor 벡터를 특징 점의 픽셀 값을 기준으로 0, 1 이진 값으로 나타냄.
BRIEF는 이미지 Matching 용이 아닌 메모리를 절약하기 위한 Descriptor 표현 법.

- FAST로 특징 점을 찾음
- Harris detection에서 사용하는 코너에 대한 정량적인 값을 기준으로 가장 코너일 확률이 높은 N개의 코너를 선택
- 코너로 추출 된 픽셀을 중심으로 윈도우를 형성하고 중심에 있는 코너로부터 Intensity Centroid를 계산하여 그 방향이 코너의 방향성을 대변
- BRIEF는 방향에 대한 정보를 가지지 않으므로 방향성을 가지는 steer-BRIEF에서 분산 값이 작은 steer-BRIEF를 대신하여 높은 분산 값을 가지는 rBRIEF 생성
- 이미지 Matching 할 땐 multi-probe LSH(Locality Sensitive Hashing)사용



- **Structure from Motion**

: 연속적인 카메라 영상들로부터 3차원 구조와 상대적인 카메라 자세를 구하는 방법



- **Optimization based SLAM**

: 새로운 관측치와 지도 간의 대응관계를 찾아냄으로써 센서 데이터 기반의 Problem의 제약조건을 찾아내고, 일관된 지도를 만들기 위하여 구한 제약 조건을 바탕으로 위치와 지도를 계산

Ex-1) BA(Bundle Adjustment)

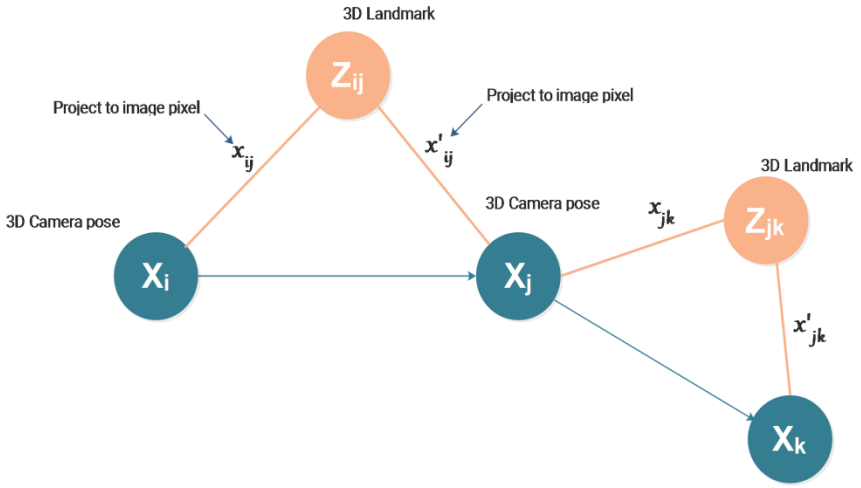
- Vision 분야에서 3D Reconstruction에서 사용
- Levenberg Marquardt 알고리즘1), Key frame, RANSAC 등 적용
- Heavy 해질 수 있는 단점 존재 -> local optimization 방법 이용

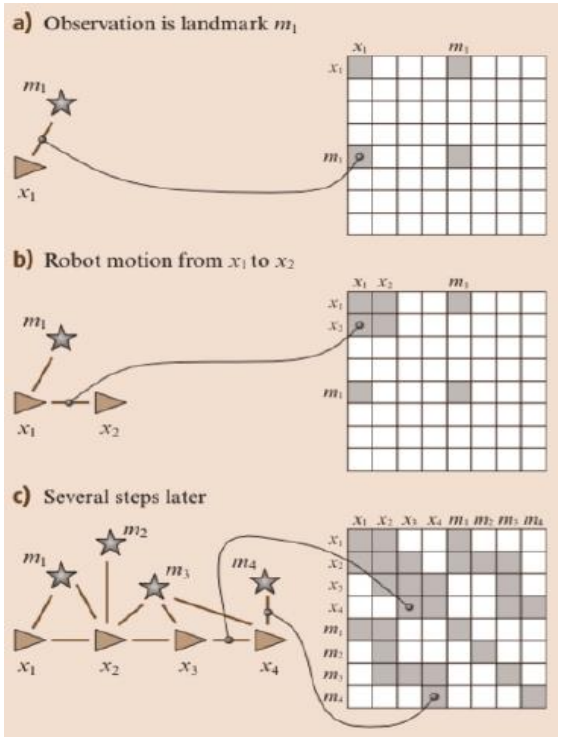
---

1) Levenberg Marquardt 알고리즘
: 비선형 최소 자승 문제를 푸는 가장 대표적인 방법
Reprojection error를 줄이고 신뢰성이 가장 높은 카메라, landmark 위치 제공

Ex-2) Graph based SLAM

: Bayesian SLAM을 Graphic하게 나타낸 방법으로써 최적화 단계에서 vehicle과 같은 platform과 landmarks 사이의 관계를 나타낸 행렬을 쉽게 만들고 사용할 수 있음

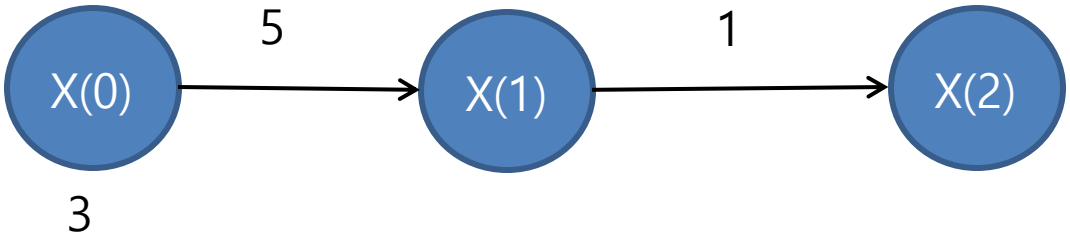




- 1-D Linear Graph SLAM

  - Movement Constraint: $X(t) = X(t-1) + D(t)$
  - Landmark Measurement Constraint: $\mathrm{L}(t) = X(t) + M(t)$

Ex) X(0) = -3, X(1) = X(0) + 5, X(2) = X(1) + 1

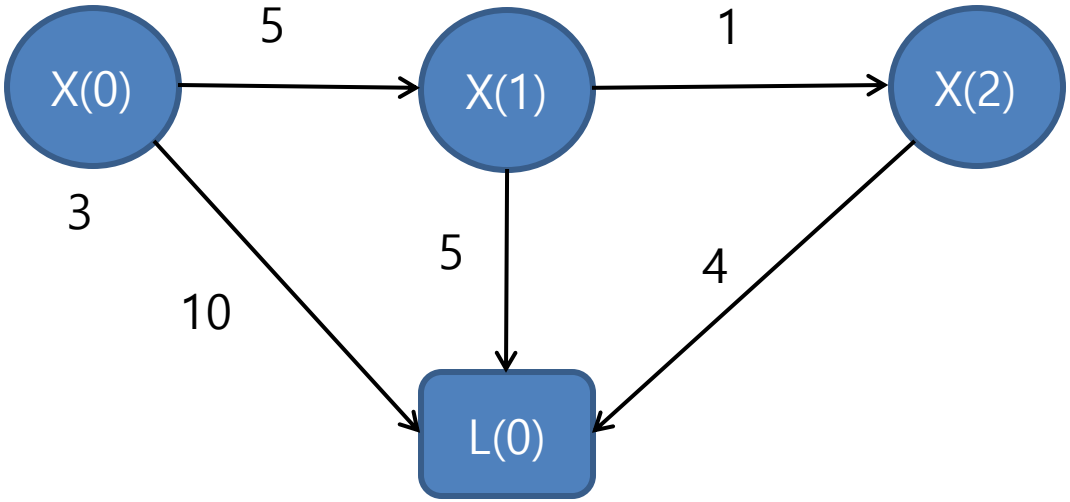


$$\begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} X(0) \\ X(1) \\ X(2) \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \\ 1 \end{pmatrix} \qquad X = \begin{pmatrix} 3 \\ 8 \\ 9 \end{pmatrix}$$

- 1-D Linear Graph SLAM

  - Adding Landmark $L(0)$

    Ex) X(0): L(0) at 10, X(1): L(0) + 5, X(2) = L(0) + 4



$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} X(0) \\ X(1) \\ X(2) \\ L(0) \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \\ 1 \\ -10 \\ -5 \\ -4 \end{pmatrix} \qquad X = \begin{pmatrix} 3 \\ 8 \\ 9 \\ 13 \end{pmatrix}$$

- 1-D Linear Graph SLAM

  - For over-determined equation(m > n), use **pseudo inverse**

$$\boldsymbol{A} * \boldsymbol{X} = \boldsymbol{B}$$
$$\boldsymbol{A^T} * \boldsymbol{A} * \boldsymbol{X} = \boldsymbol{A^T} * \boldsymbol{B}$$
$$\boldsymbol{X} = (\boldsymbol{A^T} * \boldsymbol{A})^{-1} * \boldsymbol{A^T} * \boldsymbol{B}$$
$$(\boldsymbol{A^T} * \boldsymbol{A})^{-1} * \boldsymbol{A^T}\text{: } \textbf{Pseudo inverse}$$

: $\boldsymbol{A^T} * \boldsymbol{A}$ 는 미지수 만큼의 방정식만 남기고 최적의 해를 고려함.

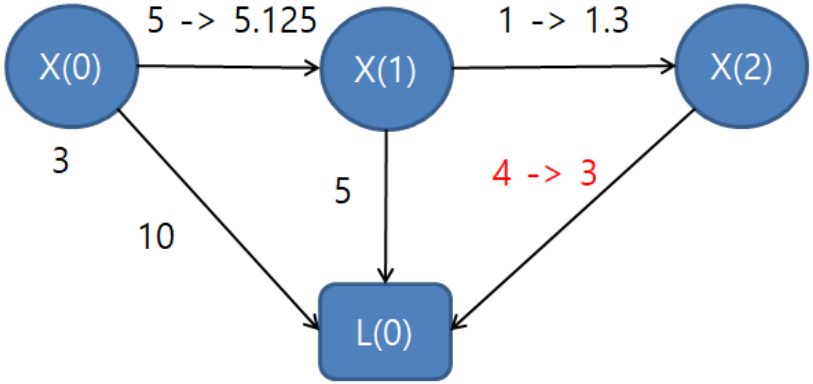
  - For python code

```
import numpy as np
import numpy.linalg as lin

A = np.array([[1, 0, 0], [-1, 1, 0], [0, -1, 1]])
B = np.array([[3], [5], [1]])
E = np.array([[1, 0, 0, 0], [-1, 1, 0, 0], [0, -1, 1, 0], [1, 0, 0, -1], [0, 1, 0, -1], [0, 0, 1, -1]])
F = np.array([[3], [5], [1], [-10], [-5], [-4]])

C1 = lin.inv(A)
C2 = np.dot(C1, B)

D1 = lin.pinv(E)
D2 = np.dot(D1, F)
print(C2)
print(D2)
```

- 1-D Linear Graph SLAM

  - Inconsistent measurement Solution

    Ex) X(0): L(0) at 10, X(1): L(0) + 5, X(2) = L(0) + 3



$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} X(0) \\ X(1) \\ X(2) \\ L(0) \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \\ 1 \\ -10 \\ -5 \\ -3 \end{pmatrix}$$

$$X = \begin{pmatrix} 3 \\ 8.125 \\ 9.5 \\ 12.875 \end{pmatrix}$$

$$L(0) = X(2) + 3.375$$

X(2)에서 L(0)를 관측하는데 오차가 발생하였으므로 이러한 에러를 줄이기 위해 행렬 식에서는 전체적으로 에러를 줄이기 위해 X 값이 변경됨

- 1-D Linear Graph SLAM

  - If we know something about how confident a measure is,
    we can include that in the computation
  - Weight matrix is diagonal matrix and It's element is 1/variance

$$\boldsymbol{X} = (\boldsymbol{A}^{\boldsymbol{T}} * \boldsymbol{W} * \boldsymbol{A})^{-\mathbf{1}} * \boldsymbol{A}^{\boldsymbol{T}} * \boldsymbol{W} * \boldsymbol{B}$$

Ex) X(0): L(0) at 10, X(1): L(0) + 5, X(2) = L(0) + 3,
X(2) measurement variance 0.1 -> Weight element 1/0.1 = 10

$$\boldsymbol{W} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{pmatrix}$$

- 1-D Linear Graph SLAM

  - Results

$$X = \begin{pmatrix} 3 \\ 8.1886 \\ 9.7547 \\ 12.811 \end{pmatrix}$$

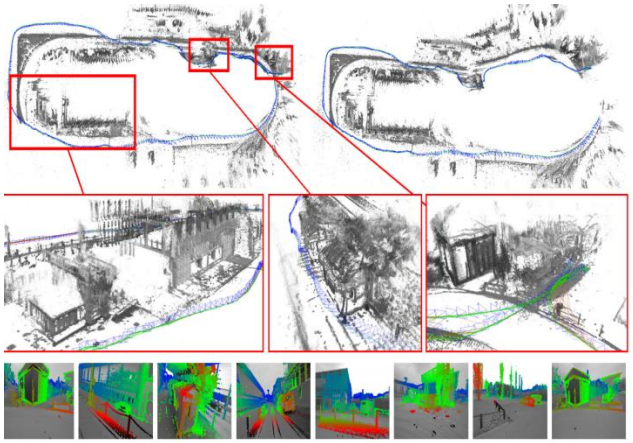
$L(0) = X(2) + 3.061$: 가중치가 커지면서 측정 값 3에 가까워짐

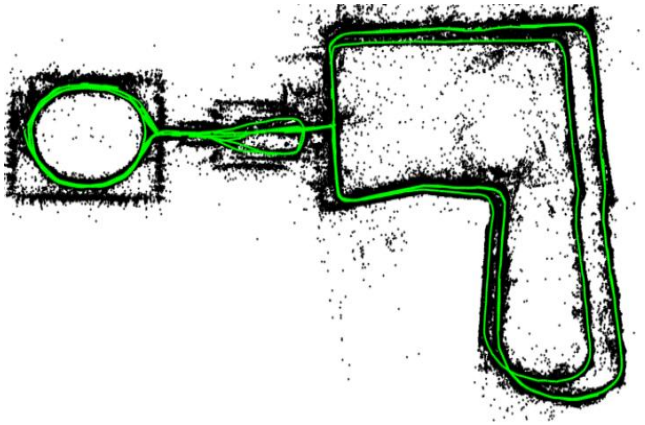
  - For python code

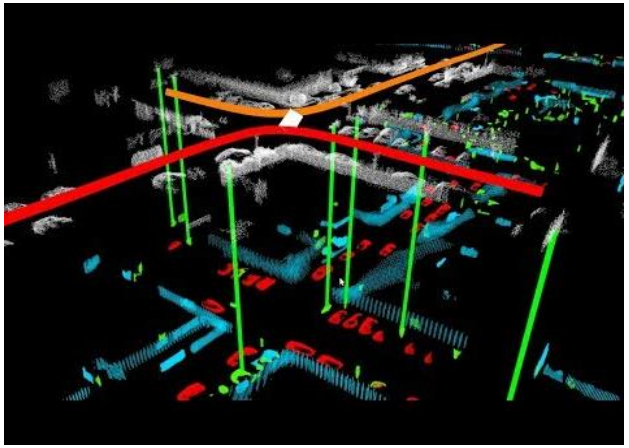
```
import numpy as np
import numpy.linalg as lin

A = np.array([[1, 0, 0, 0], [-1, 1, 0, 0], [0, -1, 1, 0], [1, 0, 0, -1], [0, 1, 0, -1], [0, 0, 1, -1]])
B = np.array([[3], [5], [1], [-10], [-5], [-3]])
W = np.array([[1, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 0, 10]])

AtwA = lin.inv(np.dot(np.dot(A.T, W), A))
AtwB = np.dot(np.dot(A.T, W), B)

X = np.dot(AtwA, AtwB)

print(X)
```

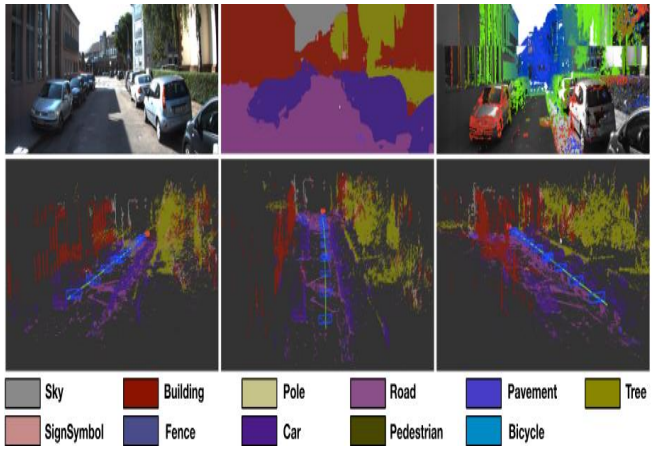
- **Modern Visual SLAM**

- Large Scale
- Sparse / Dense
- Real time
- Lidar
- Batch

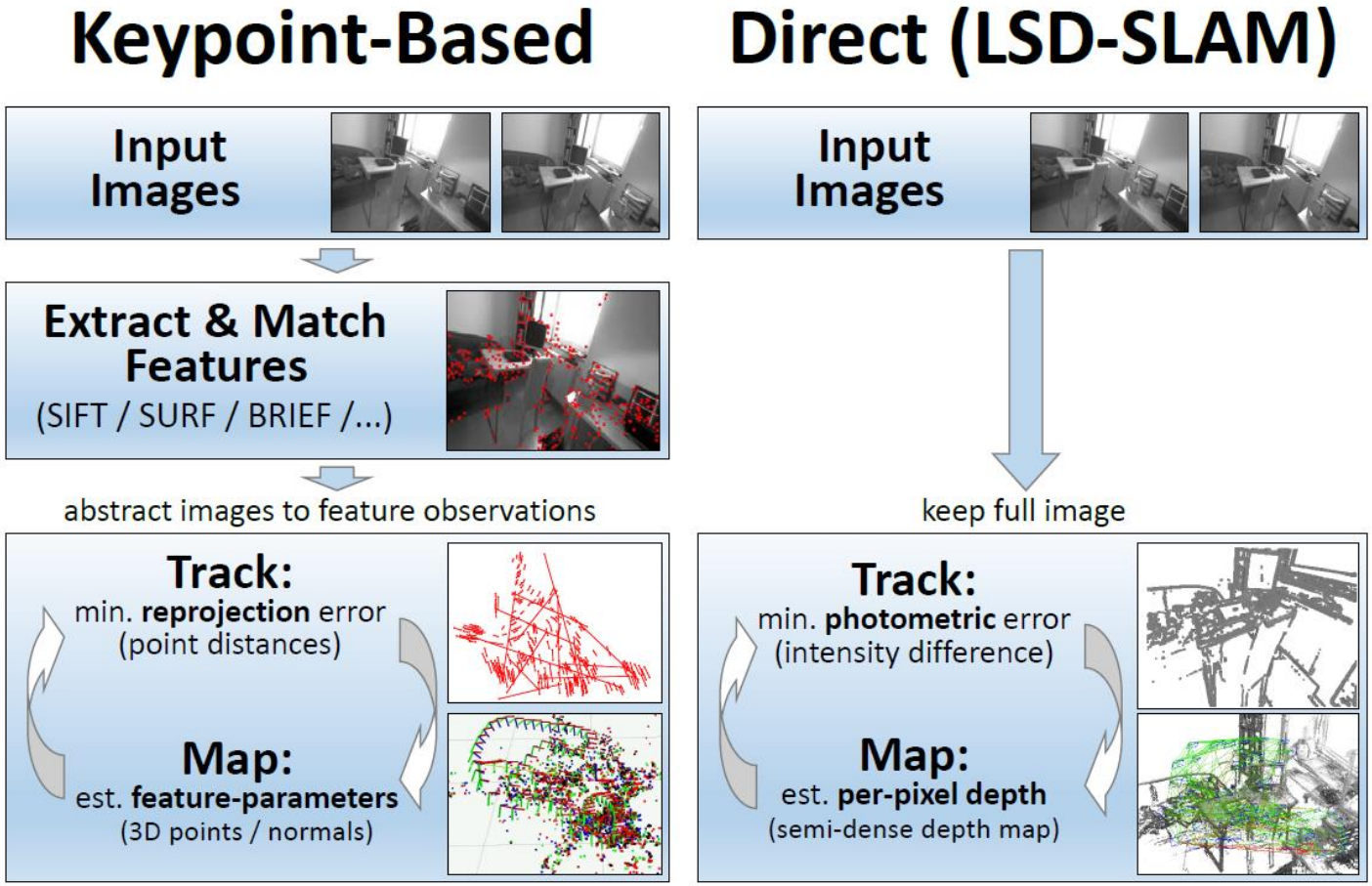








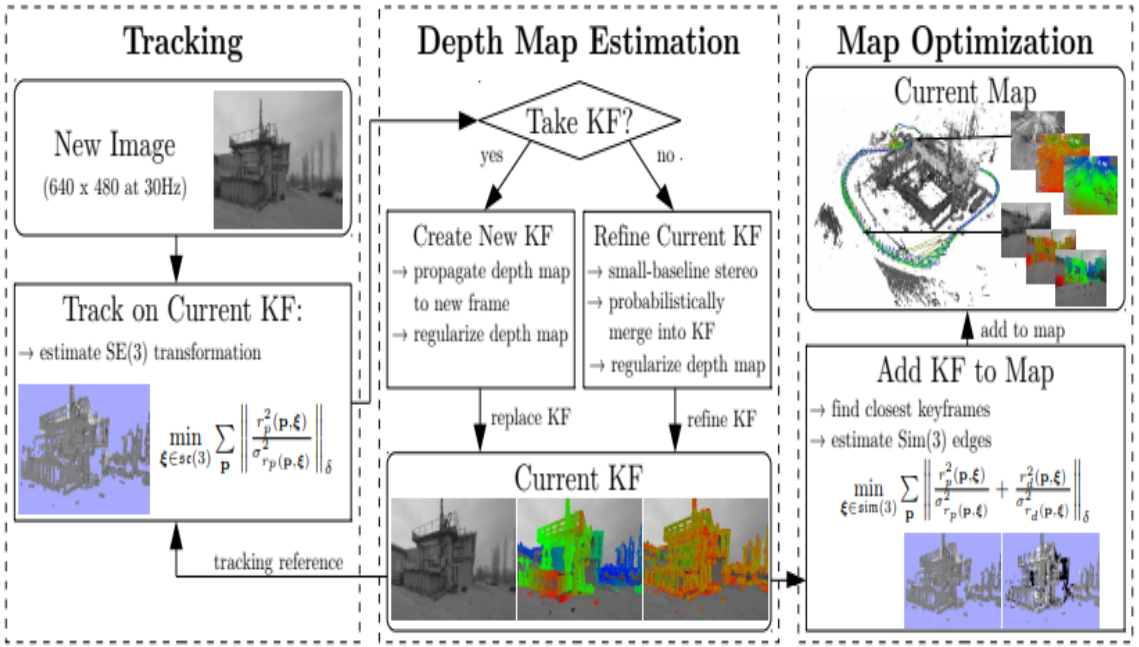
‘Feature-Based SLAM’ VS ‘Direct SLAM’



• LSD-SLAM(Large Scale Direct Monocular)

- Feature
- Large Scale
- Fully Direct (feature less)
- Real time
- CPU

- Key points
- Tracking
- Depth Map Estimation
- Map Optimization

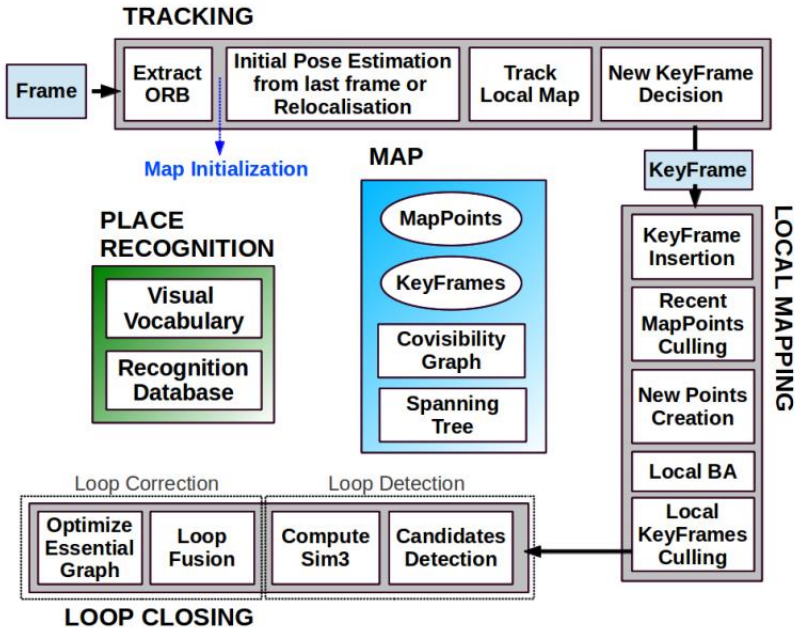


LSD-SLAM: Large-Scale Direct Monocular SLAM, Jakob Engel and Thomas Sch¨ops and Daniel Cremers, ECCV, 2014
URL: https://github.com/tum-vision/lsd_slam

• ORB-SLAM(Oriented FAST and Rotated BRIEF)

- Feature
- Feature Based
- Real time
- Small & Large/Indoor & Outdoor
- CPU

- Key points
- Tracking(+Initialize Map)
- Local Mapping
- Loop Closing

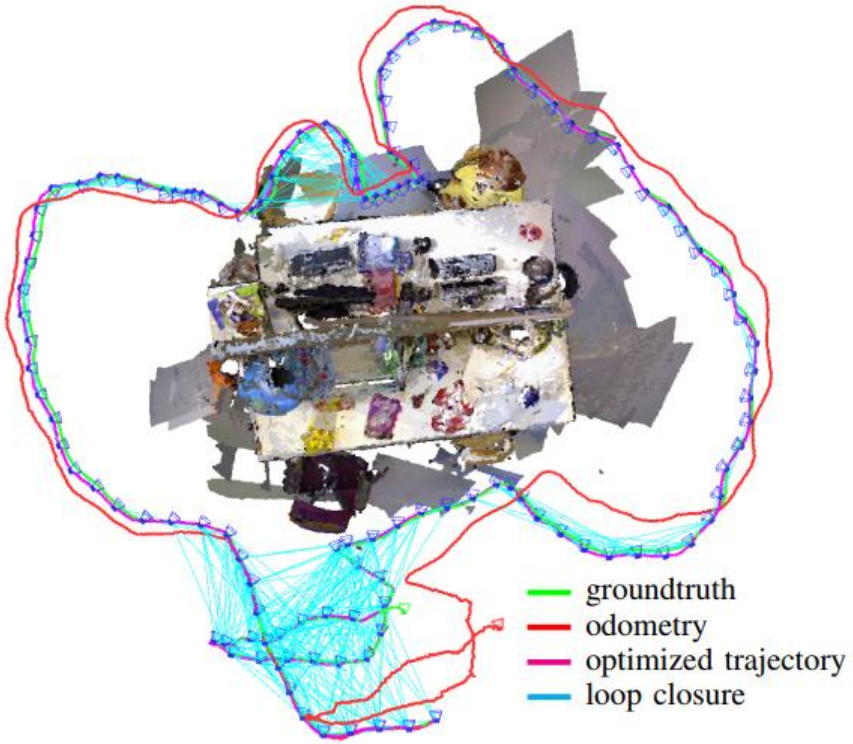


R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1 163, 2015.
URL: https://github.com/OpenSLAM-org/openslam_orbslam , https://github.com/raulmur/ORB_SLAM2

- **Dense Visual SLAM**

- Feature
- Key frame-Based
- RGB-D Camera
- ICP

- Key points
- Frame to Frame Registration
- Entropy-based method
  Using to select key frames
  validate Loop Closure



Dense Visual SLAM for RGB-D Cameras (C. Kerl, J. Sturm, D. Cremers), *In Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*, 2013
URL: https://github.com/tum-vision/dvo_slam

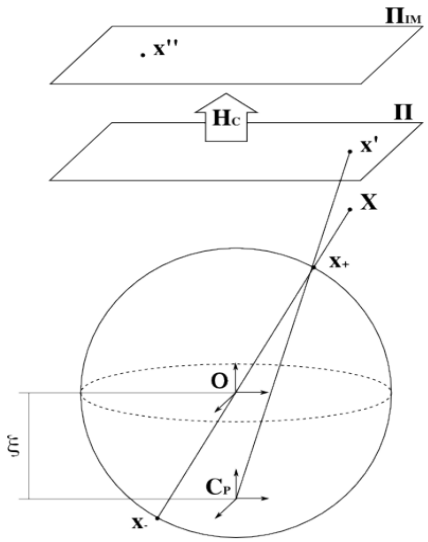
3. Omnidirectional Camera

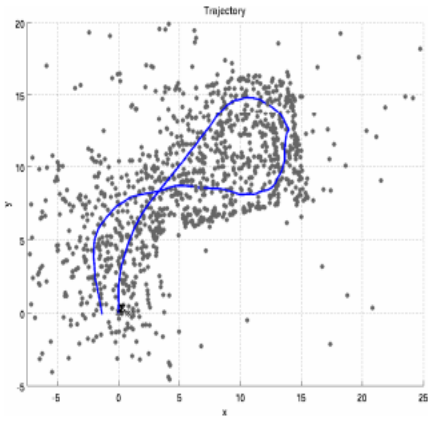
- **Comparison of omnidirectional and conventional monocular systems for visual SLAM**

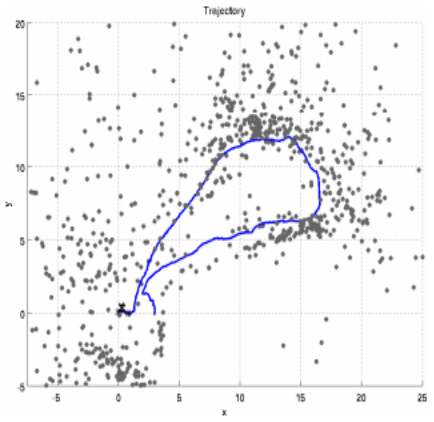
(Alejandro Rituerto1 , Luis Puig2 , J. J. Guerrero), 10th OMNIVIS with RSS, 2010

Key points
- Spherical Camera model
- EKF-Filter
- SIFT
- Outdoor

"Superiority of the omnidirectional system for the estimation of trajectory and orientation, 3D reconstruction"







- **Bearing Only FastSLAM Using Vertical Line Information from an Omnidirectional Camera**

(Mahisorn Wongphati, Nattee Niparnan and Attawith Sudsang), International Conference on Robotics and Biomimetics Bangkok, Thailand, February 21 - 26, 2009

Key points
- FastSLAM (Particle Filter)
- Bearing Information
- Vertical Line
- Panoramic

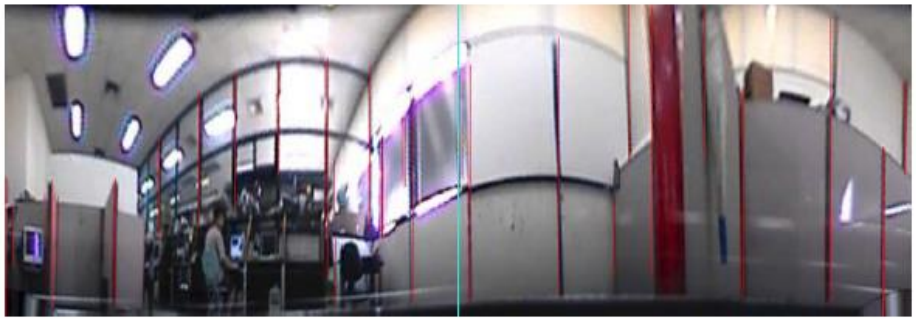
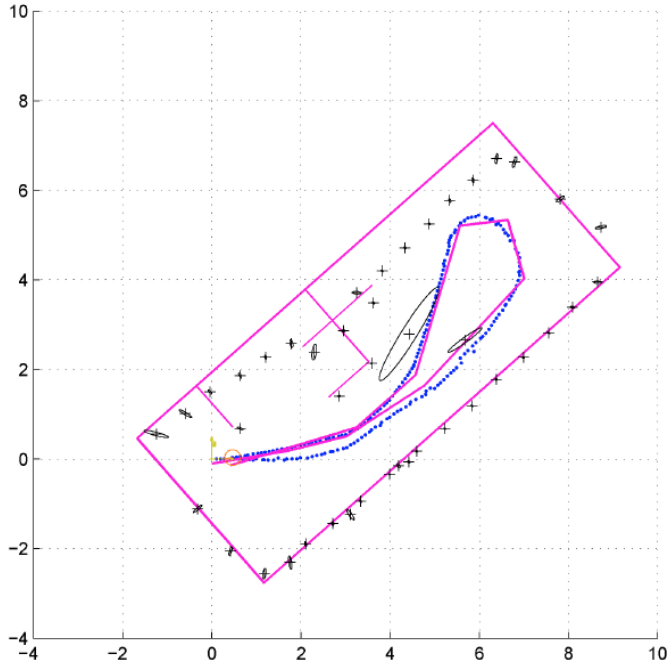


Fig. 3. Vertical lines extract using our method (red line)



- **Large-Scale Direct SLAM for Omnidirectional Cameras**

(D. Caruso, J. Engel, D. Cremers), *In International Conference on Intelligent Robots and Systems (IROS)*, 2015.

URL: https://vision.in.tum.de/data/datasets/omni-lsdslam

Key points
- Extension of LSD SLAM
- Omnidirectional Camera Unified Model
- Inverse Distance instead of Depth
- Direct Image Alignment
- Distorted Stereo Matching