Voxel-SLAM

담당자	대현 대현 표
진행 상태	진행 중
프로젝트	
태그	SLAM imu 논문 오픈소스 이론

- Paper Info
  - Title
    - Voxel-SLAM: A Complete, Accurate, and Versatile LiDAR-Inertial SLAM System
    - Arxiv에만 올라온 상태
  - author
    - Zheng Liu, Haotian Li, Chongjian Yuan, Xiyuan Liu, Jiarong Lin, Rundong Li, Chunran Zheng, Bingyang Zhou,
      Wenyi Liu, and Fu Zhang
  - 소속
    - HKU-MARS(Univ. of Hong Kong)
  - Github
    - https://github.com/hku-mars/Voxel-SLAM / GPL-2.0 license(상업적 목적으로 사용·수정·배포 가능, 단 전체 소스코드를 공개하고, 동일한(=GPL-2.0) 라이선스를 적용)

▼ 논문 전문 번역 내용

▼ Abstract

Voxel-SLAM은 **LiDAR-Inertial SLAM** 시스템으로, **short-term·mid-term·long-term·multi-map** 네 가지 데이터 연관성을 모두 활용하여 **실시간 위치 추정**과 **고정밀 맵핑**을 동시에 달성한다. 본 시스템은 **Initialization, Odometry, Local Mapping, Loop Closure, Global Mapping**의 **5개 모듈**로

구성되며, 이들 모두가 **Adaptive Voxel Map**이라는 동일한 맵 표현 방식을 사용한다.

초기화(Initialization)는 정확한 초기 상태 추정과 일관된 로컬 맵을 제공하여, **높은 동적(initial motion) 상황**에서도 시스템이 즉시 시작될 수 있게 한다. **Odometry**는 short-term 데이터 연관성을 이용해 현재 상태를 빠르게 계산하고 잠재적 시스템 발산을 감지한다. **Local Mapping**은 mid-term 연관성을 이용해 최근 LiDAR 스캔을 슬라이딩 윈도로 묶어 **LiDAR-Inertial Bundle Adjustment(BA)** 로 상태와 로컬 맵을 정교화한다. **Loop Closure**는 현재 세션뿐 아니라 모든 과거 세션에서 재방문 장소를 탐지한다. **Global Mapping**은 효율적인 **Hierarchical Global BA** 로 전역 맵을 정밀화한다. Loop Closure와 Global Mapping 모두 long-term 및 multi-map 연관성을 적극 활용한다.

저자들은 **실내 협소 공간(손-헬드 장비), 대규모 자연 지형(항공 로봇), 도심 차량 환경** 등 3가지 대표적 장면에서 30개 시퀀스를 대상으로 최신 방법들과 포괄적 벤치마크를 수행하였다. 추가 실험을 통해 초기화의 강건성·효율성, 다중 세션 운용 능력, 그리고 변질(degenerated) 환경에서의 재정위치(re-localization) 성능을 입증하였다. 연구 성과 확산을 위해 **소스코드 전체를 공개**하였다.

▼ Introduction

3D LiDAR는 직접적(direct), 고밀도(dense), 능동적(active), 정확한(accurate) **DDAA** 방식의 깊이 계측 덕분에, 자율주행 차량[1]이나 무인 드론[2, 3]과 같은 자율 이동 로봇 분야에서 널리 사용되는 센싱 기술이 되었다.

LiDAR SLAM과 LiDAR Odometry(LO)는 LiDAR 측정을 이용하여, 로봇의 후속 과정(예: 경로 계획, 제어)에 필수적인 상태 피드백과 주변 환경의 3D 고밀도 포인트클라우드를 제공한다.

또한 LiDAR 기술 발전으로, 소형·경량·저가의 이점을 가진 **solid-state LiDAR**가 활발히 연구[4]–[6]되어 일부 LO[7, 8] 및 LiDAR SLAM 시스템[9, 10]에 적용되고 있다.

LiDAR SLAM과 LiDAR Odometry(LO)의 차이점은 명확하다. SLAM의 목표는 일관된(environment-consistent) 지도를 구축하면서 그 지도 상에서 로봇의 현재 자세를 실시간으로 추정하는 것이다. 반면, LO는 실시간 위치 추정에 집중하고 지도의 누적만 수행할 뿐 drift를 줄이기 위한 정제(refinement)는 고려하지 않는다.

ORB-SLAM3[11, 12]의 개념을 따를 때, LiDAR SLAM이 LO보다 우위에 있는 지점은 **네 가지 데이터 연관성**(data association)을 모두 활용하여 과거 측정을 재매칭·정제할 수 있다는 것이다.

(1) Short-term data association
단기 데이터 연관성은 현재 스캔을 지도에 매칭하여 가능한 한 빠르게 ego-motion(자기 위치 변위)을 추정해, 경로 계획·제어 등 후속 과정에 제공한다. 대부분의 LO/LIO 시스템[8, 13–15]은 이 단기 연관성만을 사용하며, 현재 LiDAR 스캔을 그대로 지도에 누적하여 상태나 지도를 추가로 정제하지 않으므로 누적 drift가 발생한다.

(2) Mid-term data association
중기 데이터 연관성은 최근 다수의 스캔을 한꺼번에 지도에 연결하고 이를 정제한다. 일반적으로
bundle adjustment(BA) 기법을 통해 일정 시간 구간의 상태와 로컬 지도를 동시에 최적화하여 누적 drift를 줄이고 강건성을 높인다.

(3) Long-term data association
장기 데이터 연관성은 이전에 수집된 모든 LiDAR 스캔을 연관시켜 전역 지도(global map)의 일관성을 확보한다. 반복 방문 장소(loop) 탐지 후 **pose-graph optimization(PGO)**으로 드리프트를 보정하고, 더 나아가 전역 BA로 정확도를 향상할 수 있다.

(4) Multi-map data association
다중 지도 데이터 연관성은 여러 맵 세션(map session)을 서로 연관짓는다. 서로 다른 세션(수집 시점이 다름 혹은 LiDAR 변형으로 인한 재초기화 등) 간 지도를 정합하여 하나의 통합 맵을 만든다. Long-term 연관성과 유사한 방법을 사용한다.

본 연구에서는 이 네 가지 데이터 연관성을 모두 적극적으로 활용하여 **Voxel-SLAM**을 제안한다. Voxel-SLAM은 **Initialization, Odometry, Local Mapping, Loop Closure, Global Mapping**의 다섯 핵심 모듈로 구성된다. 시스템 전 단계에서 동일한 지도 표현인 **adaptive voxel map**을 사용한다. Voxel-SLAM은 **BALM2**[16]의 효율적인 LiDAR BA, **HBA**[17]의 글로벌 매핑 방법, **BTC**[18]의 장소 인식(place recognition), 그리고 **VoxelMap**[19] 기반 맵 표현 위에 구축되었다.

Voxel-SLAM의 주요 기여점은 다음과 같다:

(1) 단일 지도 형식 통일

모든 모듈에서 동일한 adaptive voxel map을 사용하여 다양한 작업(초기화·오도메트리·로컬/루프/글로벌 매핑)에 필요한 feature를 일관되게 제공한다.

(2) 견고하고 빠른 Initialization

시스템은 1초 분량의 짧은 데이터만으로 초기화(또는 divergence 후 재초기화)할 수 있으며, 정지·동적 상태 모두에서 동작한다. 초기화 과정은 정확한 상태와

일관된 초기 로컬 맵을 후속 모듈에 제공한다.

(3) 효율적인 Local Mapping

Sliding window(10 스캔) LiDAR-Inertial BA로 로컬 voxel map과 상태를 정제하여 정확도·강건성을 향상시킨다. Local mapping은 10 Hz Odometry와 동일 속도로, 연산 자원이 제한된 로봇 온보드 컴퓨터에서도 실시간으로 실행 가능하다.

(4) 다중 세션 Loop Closure

현재 세션뿐 아니라 과거 세션에서도 루프를 검출하고, 관련 scan pose를 전역적으로 최적화한다.

(5) 단일·다중 세션 모두에 대한 고효율·고정밀 Global Mapping

계층적 BA(hierarchical BA)를 활용하여 scan pose와 지도 일관성을 효율적으로 전역 최적화한다.

(6) 네 가지 데이터 연관성의 완전 활용

Odometry와 Local Mapping 단계에서 단기·중기 연관성을, Loop Closure와 Global Mapping 단계에서 장기·다중 지도 연관성을 활용함으로써, 실시간성과 전역 지도 일관성을 동시에 달성한다.

▼ Related Work

- LiDAR(-Inertial) Odometry and SLAM
  - **LOAM** [20]은 초기 LiDAR odometry 및 mapping 프레임워크로서 이후 연구에 큰 영향을 미쳤다. LOAM은 feature extraction, scan-to-scan odometry, scan-to-map mapping 세 개의 모듈로 구성된다. 각 스캔 라인의 국소 평활성(local smoothness)을 이용해 planar 및 edge feature point를 추출하여 자세 추정의 계산 부담을 줄인다. 이후 odometry 모듈은 이러한 feature point를 사용하여 현재 스캔을 이전 스캔과 정합시켜 실시간 상대 자세를 추정한다. 정밀도를 높이기 위해 mapping 모듈은 과거 feature point를 누적하여 k-d tree를 구축하고, point-to-plane(또는 edge) 거리를 비용으로 삼아 odometry 결과를 정제한다. 그러나 k-d tree를 재구성해야 하므로 mapping 모듈은 odometry(10 Hz)보다 느린 1 Hz로만 동작할 수 있다.
  - **LeGO-LOAM** [21]은 LOAM의 프레임워크를 따르면서 feature extraction 단계에 ground segmentation을 도입하고 loop closure를 추가하여 장기 drift를 완화하였다.
  - **LOAM-Livox** [7]는 이 프레임워크를 solid-state LiDAR에 맞추어 변형하였다. Solid-state LiDAR의 좁은 FoV와 비반복 스캔 패턴으로 인해 두

연속 스캔 간 상호 대응점이 매우 적다. 따라서 LOAM-Livox는 odometry와 mapping 모두에서 scan-to-map registration만 사용한다. 이는 odometry 정확도를 높이지만 매 스캔마다 k-d tree를 재구축해야 하므로 계산 부하가 증가한다. IMU를 통합하면 더 나은 초기 자세 제공, LiDAR 스캔의 motion distortion 보정, 추정 강건성 향상을 얻을 수 있다.

- **LIOM** [22]은 VINS-Mono [23]에서 영감을 받아 개발된 최초의 공개형 tightly-coupled LiDAR-Inertial Odometry(LIO) 중 하나이다. 그러나 효율적인 BA 방법이 없어 LIOM의 batch optimization은 실시간으로 실행하기엔 너무 느리다.
- **LiLi-OM** [24]은 LIOM과 유사하게 local map과 sliding-window optimization을 사용한 tightly-coupled LIO이며, solid-state LiDAR에 적응하고 ICP 기반 loop closure를 도입하였다. 실시간성을 확보하기 위해 LiLi-OM은 작은 optimization window를 사용한다.
- **LIO-SAM** [25]은 LIOM의 효율이 낮은 local map을 버리고, LiDAR odometry 결과를 factor graph에 직접 투입하여 IMU pre-integration[26]과 함께 최적화한다. 이에 따라 LIO-SAM은 더 큰 sliding window를 유지할 수 있고 GNSS와 같은 다른 센서와도 쉽게 융합된다.
- **LINS** [27]는 여러 상태를 동시에 최적화하지 않도록 filter-based 방법을 도입하였다.
- **FAST-LIO** [8]는 Kalman gain 계산 차원을 줄이기 위해 새로운 Kalman gain을 제안하였다.
- **FAST-LIO2** [13]는 feature extraction을 제거하고 incremental k-d tree를 설계하였다.
- **LTA-OM** [10]은 FAST-LIO2에 loop closure를 추가하고 STD place recognition descriptor [28]를 사용하였다.
- **Faster-LIO** [14]는 FAST-LIO2의 incremental k-d tree를 병렬 sparse incremental voxels로 대체하여 효율을 높였다.
- **Point-LIO** [15]는 point-by-point LIO 프레임워크로 고주파수 odometry 출력과 급격한 움직임·IMU 포화 상황에서도 높은 강건성을 보인다.
- 최근에는 **LiDAR bundle adjustment(BA)**를 LIO에 결합하여 drift를 줄이려는 시도[29, 30]도 있었지만 아직 공개 구현은 없다.

    - 기존 오픈소스 LiDAR(-Inertial) SLAM 시스템과 비교할 때, **Voxel-SLAM**은 (i) 동적 초기 상태에서도 시작할 수 있는 Initialization, (ii) LiDAR-Inertial BA를 활용하는 Local Mapping, (iii) 다중 세션 루프 검출 Loop Closure, (iv) 글로벌 정밀화를 위한 Global Mapping 모듈을 모두 포함한다. Voxel-SLAM은 단기·중기·장기·다중 지도 네 가지 데이터 연관성을 모두 활용해 실시간 동작과 전역 지도 일관성을 동시에 달성한다.

- LiDAR Scan Registration Scheme
    - LiDAR 포인트클라우드 정합(registration) 방법은 **pairwise registration**과 **multiview registration** 두 가지로 나뉜다.
    - Pairwise registration은 단일 포즈만 추정하며, LO 및 대부분의 LIO에서 주로 사용된다. LiDAR 스캔은 희소(sparse)하므로 LOAM 및 후속 연구들은 point-to-point 대신 point-to-plane(edge) registration을 사용하여 최신 스캔을 기존 points map에 매칭한다. 그러나 points map 방식은 새 스캔 정합을 위해 plane·edge를 재구성해야 하므로 추가 시간이 필요하다. 이를 해결하기 위해 **SuMa** [31], **LIPS** [32], **VoxelMap** [19]은 plane map을 사용해 각 포인트를 현재 스캔의 대응 평면과 매칭함으로써 효율과 정확도를 동시에 높였다.
    - Pairwise registration만 사용하면 누적 drift가 발생하므로, multiview registration은 여러 스캔을 동시에 정합하여 이를 해결한다.
    - **EigenFactor(EF)** [33]는 각 포인트가 해당 평면에 갖는 거리를 비용으로 두고, 이를 covariance 행렬의 최소 고유값으로 전환하여 gradient 기반으로 최적화한다. 하지만 eigenvalue 류 비용은 2차 특성 때문에 수렴 속도가 느리다.
    - **Plane Adjustment(PA)** [34]는 visual BA를 참고해 pose와 plane parameter를 동시 변수로 하고, Schur complement로 plane 차원을 줄인다. 그러나 plane 수가 많을수록 반복(iteration) 횟수가 늘고 plane parameter에 특이성(singularity)이 존재한다.
    - **BAREG** [35]는 휴리스틱 penalty term을 비용함수에 추가해 수렴은 빠르지만 실제 noisy 측정에서는 map consistency를 해칠 수 있다.
    - **BALM** [36]은 2차 도함수 기반 solver와 기하적 feature(plane·line)의 해석적(analytic) 해결을 제시해 훨씬 적은 단계로 수렴한다. 다만 BALM은 Jacobian·Hessian을 계산하기 위해 모든 포인트를 열거(enumerating)해야 해서 계산 복잡도가 높다.
    - **BALM2** [16]는 이를 해결하기 위해 (i) point cluster로 raw point를 압축 파라미터화하고, (ii) Hessian·Jacobian의 해석형을 유도하여 빠른 수

렴과 높은 정확성을 달성했다. BALM2를 기반으로 Liu et al.은 대규모 글로벌 매핑을 위한 **Hierarchical Bundle Adjustment(HBA)** [17] 프레임워크를 제안하였다.

    - **Voxel-SLAM**은 pairwise와 multiview 정합을 모두 활용한다. Odometry 단계에서는 VoxelMap [19] 기반 plane map으로 빠른 scan-to-map 정합을 수행한다. Initialization과 Local Mapping에서는 BALM2 기반 LiDAR(-Inertial) BA를 채택해 효율과 정확성을 동시에 확보한다. 또한, 전역 정확도를 위해 Global Mapping 모듈에 효율적인 HBA [17]를 포함한다.

- LiDAR-Based Place Recognition
    - **Magnusson et al.** [37]은 NDT에서 영감을 얻어 포인트클라우드를 정규 격자(grid)로 분할하고 각 cell을 histogram matrix로 인코딩한 후, 두 matrix를 매칭해 루프 클로저를 탐지하였다.
    - **M2DP** [38]는 3D 포인트클라우드를 다수의 2D 평면으로 투영하고, 각 평면의 밀도 서명(density signature)을 생성하여 SVD의 좌·우 특이 벡터를 descriptor로 사용한다.
    - **SegMatch** [39]는 포인트클라우드를 세그먼트화하여 semantic feature를 추출하고, 이를 이용해 루프를 검출한다.
    - **Scan Context** [41]는 shape context[40]에서 착안해 새로운 공간적 descriptor를 제안하여 3D 포인트클라우드를 2.5D 정보 행렬로 인코딩하였다.
    - **Scan Context++** [42]는 Scan Context의 lateral 불변성 부족과 brute-force 검색 문제를 해결하기 위해 generic descriptor와 sub-descriptor 기반 빠른 검색 방식을 제시하였다.
    - **BoW3D** [44]는 Visual SLAM의 BoW에 착안하여 LiDAR 포인트의 Link3D feature로 BoW 딕셔너리를 구축한다.
    - **CNN/딥러닝** 기반 방법(LCDNet [45], LoGG3D-Net [46], LocNet [47])은 포인트클라우드에서 local feature를 학습해 global descriptor로 부호화한다. 그러나 이러한 학습 기반 방법은 훈련 데이터에 민감하고 일반화 능력이 제한적이어서 LiDAR 종류나 환경이 바뀌면 재학습이 필요하다.
    - 이에 Yuan et al.은 견고한 이진 특징과 triangle descriptor 매칭을 결합한 **BTC(Binary Triangle Combined)** descriptor [18, 28]를 제안하였

다. 이 방식은 시야(viewpoint)에 강인하고 LiDAR 종류에 쉽게 적응하며 6-DoF 상대 자세도 빠르게 추정할 수 있다.

- 정확도와 효율을 고려해 Voxel-SLAM은 루프 클로저 검출에 BTC[18]를 채택했다. 앞서 소개한 plane-map 기반 scan-to-map 정합이나 (hierarchical) LiDAR BA처럼, BTC 역시 voxel map을 사용해 plane을 검출하고 descriptor를 추출한다. 따라서 adaptive voxel map은 Voxel-SLAM의 Initialization, Odometry, Local Mapping, Global Mapping 등 모든 모듈에서 **통합 맵 구조**로 활용된다.

▼ System Overview

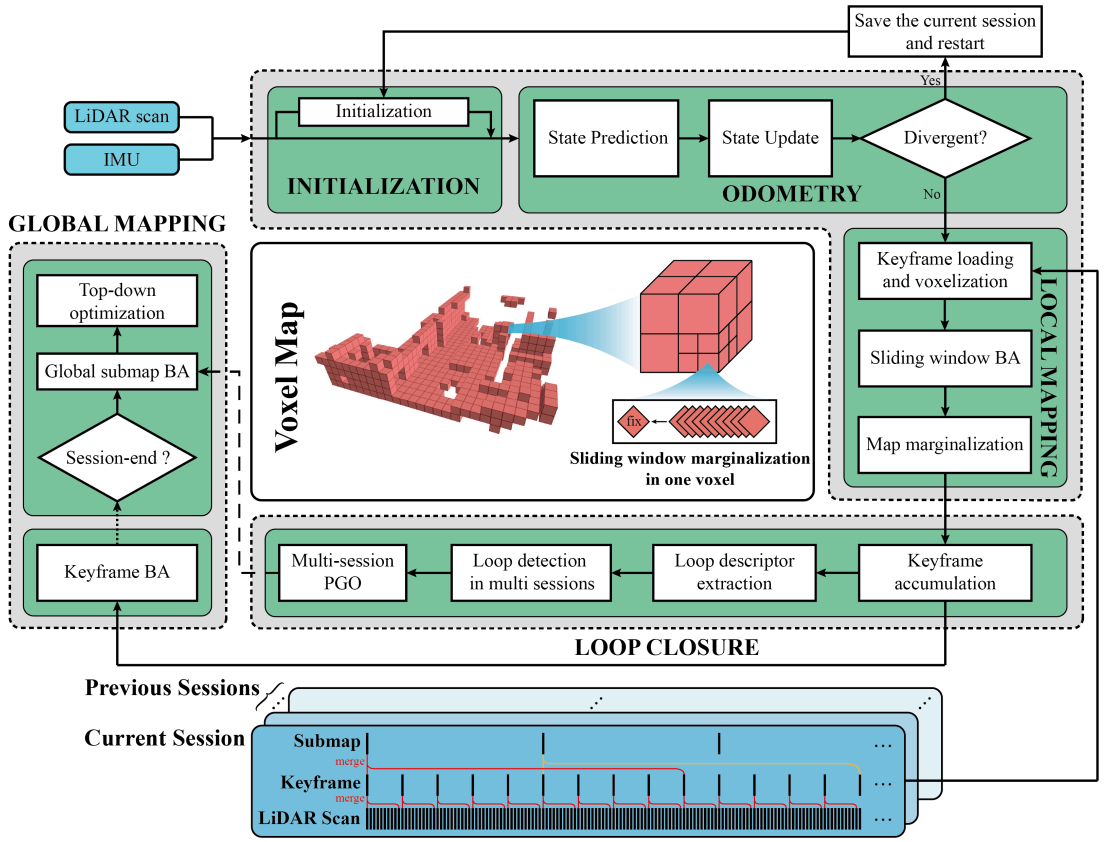

그림1. Voxel-SLAM의 개요. 초록색 부분은 initialization, odometry, local mapping, loop closure, global mapping으로 구성된 Voxel-SLAM의 여러 모듈을 나타낸다. 동일한 회색 점선 상자 안의 모듈들은 동일한 스레드에서 실행된다. 하단의 파란색 부분은 다중 세션을 위한 data pyramid이며, 중앙의 빨간색 부분은 adaptive voxel map이다.

시스템 개요는 그림 1에 제시되어 있다. 그림 1의 초록색 영역은 Voxel-SLAM의 각기 다른 모듈을 나타내며, 동일한 스레드에서 실행되는 모듈들은 회색 점선 프레임으로 함께 묶여 있다. 파란색 영역은 *data pyramid*, 빨간색 영역은 각 모듈이 공유하는 **adaptive voxel map**이다.

i 번째 LiDAR 스캔이 끝날 때의 시스템 상태를 다음과 같이 정의한다.

$$\mathbf{x}_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{p}_i & \mathbf{v}_i & \mathbf{b}_i^g & \mathbf{b}_i^a \end{bmatrix} \qquad (1)$$

여기서 Ri ∈ SO(3), pi ∈ $\mathbb{R}^3$, vi ∈ $\mathbb{R}^3$는 각각 월드 프레임에서의 IMU 회전·위치·속도를 의미한다.

월드 프레임은 최초 IMU 프레임을 기반으로 하되 z축을 중력 벡터와 정렬하며, 이는 Initialization 모듈(섹션 V-C)에서 결정된다.

bg_i ∈ $\mathbb{R}^3$와 ba_i ∈ $\mathbb{R}^3$는 IMU 바디 프레임 기준 자이로스코프 바이어스와 가속도계 바이어스이다.

- **Workflow**

  그림 1에서 보듯 Voxel-SLAM은 **세 개의 병렬 스레드**에서 동작하는 **다섯 개 모듈**로 구성되어 있다.

  모듈1: Initialization

  - 먼저 시스템은 (시작·재시작 직후 초기화되지 않았을 경우) **1초 분량**의 LiDAR 스캔과 IMU 데이터를 바탕으로 초기화 과정을 수행한다. 이 단계에서는 특수한 **LiDAR-Inertial BA** 최적화를 통해 **states of all scans, initial local map, and the gravity vector in the world frame**를 추정한다.

  모듈2: Odometry

  - 이렇게 얻은 **초기 state, local map, and gravity vector**를 이용해 **Odometry**는 LiDAR·IMU 측정을 tightly 융합하여 실시간으로 현재 state를 추정한다. 또한 지속적인 **LiDAR degeneration**으로 인한 잠재적 시스템 발산을 감지한다.

  모듈3: Local Mapping

  - 이어서 **Local Mapping**은 현재 스캔을 슬라이딩 윈도우에 포함시킨 후, **LiDAR-Inertial BA**를 통해 윈도우 내 모든 **state**와 **local map**을 동시에 정제한다. 슬라이딩 윈도우에서 가장 오래된 스캔은 **marginalized**되어 **Keyframe**으로 누적된다.

  모듈4: Loop Closure

  - 이렇게 생성된 **Keyframe**은 **Loop Closure** 모듈에서 **Loop descriptors**를 추출하고 현재 및 이전 세션 모두에서 루프를 탐지하는 데

사용된다. 루프가 성공적으로 탐지되면 현재 및 이전 세션에서 관련된 pose들은 **Pose Graph**를 구성해 최적화한다.

모듈5: Global Mapping

- Pose-Graph Optimization(PGO) 이 끝나면 **Global Mapping**이 Keyframe BA를 수행하고 Submap에 Keyframe을 실시간으로 병합한다. 세션 종료 신호가 들어오면 Global Mapping은 모든 Submap에 대해 Global BA와 top-down 최적화를 실행하여 모든 스캔의 정확한 pose를 계산한다.

- **Data Pyramid**

Input 으로 들어오는 LiDAR 포인트클라우드 시퀀스는 **세 단계 피라미드 구조**로 보관된다.

Bottom level

- 원시(raw) LiDAR 스캔으로, 보통 10 Hz로 센서에서 직접 수집된다. 이 스캔들은 Odometry·Local Mapping에서 실시간 state estimation에 사용된다.

Middle level

- LiDAR 스캔 **10장**을 묶어 하나의 **Keyframe**을 만들며, 이는 Loop Closure 단계에서 loop descriptor 추출에 활용된다.

Top level

- Keyframe **10장**을 다시 합쳐 **Submap**이 생성되며, 이는 Global Mapping 모듈에서 사용된다.
- Scan → Keyframe, Keyframe → Submap 병합은 LiDAR BA로 수행된다. 각 병합 윈도우에서는 첫 Scan(또는 Keyframe)을 기준으로 모든 Scan(또는 Keyframe) pose를 동시에 최적화한다.
- Keyframe → Submap 병합 시에는 윈도우 크기 10, 보폭(stride) 5인 슬라이딩 윈도우를 사용한다. 이로 인해 연속된 두 윈도우가 Keyframe 5장을 공유하여 Submap 간 공시야 영역(co-visible areas)을 확장한다.
- 반면 Scan → Keyframe 병합 윈도우는 겹침이 없다(크기 10, 보폭 0). 이는 LiDAR의 긴 측정 거리로 인해 연속된 Keyframe 사이에서 이미 Scan 간 충분한 겹침이 보장되기 때문이다.

- **Adaptive Voxel map**

  Adaptive voxel map
  은 모든 모듈에서 평면(plane) 특징(feature) 추출과 multiple Scans·Keyframes·Submaps 간 특징 연관성 제공에 핵심적 역할을 한다.

  Voxel map은 환경 내 다양한 크기의 평면 특징을 유지한다. 이를 위해 공간을 균일한 크기의 *root voxel*(크기 L_r)로 분할한다. 균일한 voxel들은 **해시 테이블**로 관리되어 각 LiDAR 포인트가 위치에 따라 해당 voxel로 매핑된다. 하나의 voxel은 **multiple layer octree** 구조를 갖고, 각 leaf 노드는 하나의 평면을 나타낸다. Layer가 다르면 leaf 크기도 달라 서로 다른 크기의 평면 특징을 표현할 수 있다.

  Adaptive voxel map 구성 절차는 다음과 같다.

  - 각 Scan·Keyframe·Submap의 포인트를 위치에 따라 해당 voxel에 배치한다.
  - 만약 voxel 내 포인트들이 동일 평면이라면(공분산 행렬 최소·차소 eigenvalue의 ratio가 임계값 이하) 그 voxel을 **plane voxel**로 간주하고 LiDAR point들을 저장한다.
  - 그렇지 않으면 voxel을 재귀적으로 더 작은 sub-voxel로 분할하고, 최소 sub-voxel 크기 또는 최소 포인트 개수 조건을 만족할 때까지 반복한다.

  Adaptive voxel map은 모듈별로 다른 목적에 사용된다.

  - 첫째, LiDAR 주변 거리 L_m 내의 **local adaptive voxel map**은 **Initialization·Odometry·Local Mapping** 모듈에서 공유되어 LiDAR 스캔 state를 실시간으로 추정한다(그림 1의 Voxel map 참조).
    1. Initialization은 초기 scan들을 받아 voxel map을 초기화한다.
    2. Odometry는 현재 scan을 voxel map의 plane과 정합하여 현재 state를 추정한다.
    3. Local Mapping은 local map을 슬라이딩하며 LiDAR-Inertial BA 최적화로 현재·최근 scan state를 정제한다.
  - 이 local voxel map 외에도, 2개의 다른 adaptive voxel map 구조도 사용된다. **Loop Closure** 모듈은 keyframe에서 BTC descriptor를 추출할 때 voxel map을 사용하고, **Global Mapping** 모듈은 HBA 수행 시 다

양한 Keyframe·Submap 간 평면 특징 추출·연관에 voxel map을 사용한다.

▼ LiDAR-Inertial BA Optimization

LiDAR-inertial BA(bundle adjustment) 최적화는 초기화(Initialization)와 로컬 매핑(Local Mapping) 모듈에서 여러 스캔의 상태를 동시에 추정하는 데 사용된다.

- 필터 기반 추정 방식의 LIO 방법들[FAST-LIO, FAST-LIO2]과 비교하면, LiDAR-inertial BA는 더 긴(중기) 데이터 연관성을 활용하므로 추정 정확도와 강건성이 더 높다. 그러나 BA 방법은 연산 시간이 늘어난다는 단점이 있어, 대부분 기존 LiDAR(-inertial) 오도메트리 또는 SLAM 시스템에서 사용되기 어렵다.
- 이 문제를 해결하기 위해 **BALM2** [16]는 효율적인 LiDAR BA 기법을 제안했는데, 주요 아이디어는 다음과 같다:
  1. 평면·선과 같은 기하 특징을 해석적으로(analytically) 풀어 feature 파라미터를 최적화 변수에서 제거하여 차원을 크게 줄인다.
  2. **point cluster** 자료구조를 도입해 모든 포인트를 압축된 형태로 묶어 BA 과정에서 raw point를 일일이 열거(enumerate)하지 않는다.
  3. BA 비용 함수의 해석적 2차 도함수(Hessian)를 유도하여 효율적인 2차(Second-order) 솔버를 구현한다.
- 이 세 가지 기법 덕분에 LiDAR BA 최적화는 실시간 구현이 가능할 정도로 고속·고효율을 달성한다. **본 연구에서는 BALM2 방법을 채택하고, 여기에 IMU pre-integration [26]을 결합하여 필요하면 중력 벡터까지 함께 추정할 수 있는 LiDAR-inertial BA 최적화를 구성한다(팩터 그래프는 Fig. 2 참조).**

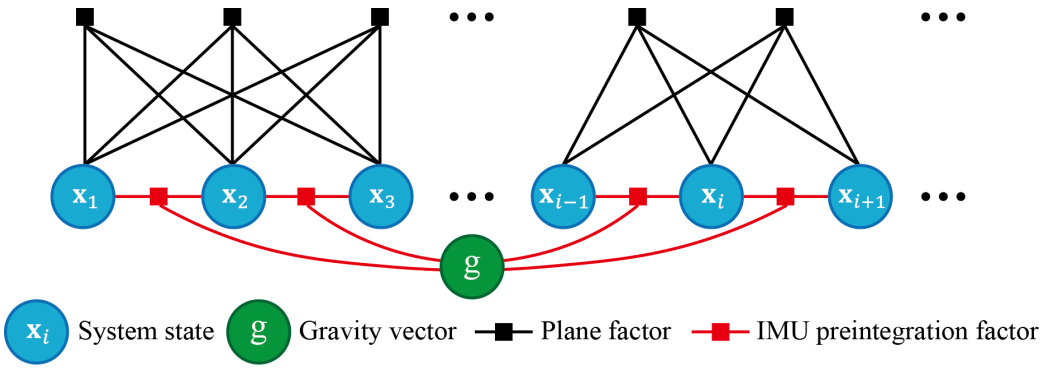

그림 2. 제안된 LiDAR-inertial BA의 Factor graph 표현.

- **비용 함수**는 다음과 같다:

$$\arg\min_{X} \left(\frac{1}{2} \sum_{i=1}^{N-1} \| r_{i,i+1}(X) \|^2_{\Sigma^{-1}_{i,i+1}} + \sum_{j=1}^{M} \lambda^j_{\min}(X) \right) \tag{2}$$

$$X = [x_1, x_2, \ldots, x_N,\ g] \tag{3}$$

- 여기서 $\mathbf{X}$는 개별 state x_i들과 중력 벡터 g로 구성된 state vector이다.
- x_i는 식 (1)에서 정의된 LiDAR 스캔 i의 자세 R_i, 위치 p_i, 속도 v_i, IMU 바이어스 b^g_i, b^a_i를 포함한다.
- $r_{i,i+1}(X)$는 i번째와 $(i+1)$번째 state 사이 IMU pre-integration 잔차(residual)이며 공분산 행렬 $\Sigma_{i,i+1}$을 갖는다.
- $\lambda^{\min}_j$는 j번째 plane feature에 대응되는 points 분포 공분산 행렬의 **최소 고유값(eigenvalue)**으로, LiDAR BA factor의 기여도라는 오류 항이다(자세한 내용은 BALM 논문 참고).
- N과 M은 각각 포함된 state 개수와 plane feature 개수를 의미한다.

- **IMU pre-integration 잔차 $r_{i,j}(i<j)$는 [26]의 정의를 그대로 따른다:**

$$\begin{aligned}
r_{i,j} &= \left[r_{\Delta R_{ij}},\ r_{\Delta p_{ij}},\ r_{\Delta v_{ij}},\ r_{\Delta b^g_{ij}},\ r_{\Delta b^a_{ij}} \right] \\
r_{\Delta R_{ij}} &= \mathrm{Log}\left(\widetilde{\Delta R}^{\top}_{ij}(b^g_i)\, R_i^{\top} R_j \right) \\
r_{\Delta p_{ij}} &= R_i^{\top}\left(p_j - p_i - v_i \Delta t_{ij} - \tfrac{1}{2} g\, \Delta t^2_{ij} \right) - \widetilde{\Delta p}_{ij}(b^g_i, b^a_i) \\
r_{\Delta v_{ij}} &= R_i^{\top}\left(v_j - v_i - g\, \Delta t_{ij} \right) - \widetilde{\Delta v}_{ij}(b^g_i, b^a_i) \\
r_{\Delta b^g_{ij}} &= b^g_j - b^g_i, \quad r_{\Delta b^a_{ij}} = b^a_j - b^a_i
\end{aligned} \tag{4}$$

- 여기서 $R_i, p_i, v_i, b^g_i, b^a_i$는 (식 1)에서 정의된 state x_i를 구성하며, $r_{i,j}$의 상태 x_i에 대한 Jacobian은 [26]과 동일하다.
- $\widetilde{\Delta R}ij^{\top}(b^g_i)$, $\widetilde{\Delta p}ij(b^g_i, b^a_i)$, $\widetilde{\Delta v}_{ij}(b^g_i, b^a_i)$는 [26]의 1차 전개(first-order expansion)를 적용한 보정된(pre-integrated) 회전·위치·속도이다.
- Δt_{ij}는 시각 i와 j 사이의 시간 간격이다.
- $\mathrm{Log} : SO(3) \to \mathbb{R}^3$는 Lie 군을 벡터 공간으로 사상하는 로그 지도(log map)이다.

- $g \in \mathbb{R}^3$는 중력 벡터이다.
- 주목할 점은, **중력 벡터를** $\mathbb{S}^2$ 상의 고정 크기 벡터로 두는 visual SLAM과 달리, **본 연구에서는 크기까지 최적화 대상으로 두기 위해** $\mathbb{R}^3$ **공간의 벡터로 취급한다는 것이다.** 이는 LiDAR 계측이 절대적 스케일 정보를 제공하므로 데이터만으로 스케일을 결정할 수 있기 때문이다. 반면, 시각 센서는 스케일 정보를 갖지 않기 때문에, 대부분의 visual SLAM 시스템[11, 23]에서는 중력 크기를 고정(예: $|g|=9.81\,m/s^2$)해야 한다.
- **(식 2) 비용 함수**는 2차(Second-order) **LM(Levenberg–Marquardt) 솔버**를 이용해 반복적으로 최적화한다. 최적화 속도를 높이기 위해 비용 함수의 Jacobian과 Hessian을 **해석적(analytic) 형태**로 유도하였다. Jacobian, Hessian, 그리고 해당 솔버의 세부 사항은 **Appendix A**에 제시돼 있다.

▼ Initialization

많은 필터 기반 LIO 시스템(Fast-lio, Fast-lio2, Faster-lio, Point-lio)은 센서가 초기에 **정지 state**라고 가정한다. 그러나 비(非)정지 초기 state(예: 시퀀스 중간에 재시작)에서는, 이러한 정지 가정이 잘못된 state를 전파하여 LiDAR 스캔의 distortion 보정과 스캔 정합(feature association)에 오류를 초래할 수 있다. 이는 초기 구간의 state estimation 정확도를 심각하게 저하시키며, 일관성 없거나 (심지어 잘못된) 초기 지도를 생성해 이후 Odometry 성능도 저하시킨다.

이러한 한계를 해결하기 위해, 우리는 **previous LiDAR-inertial BA optimization 기반의 견고하고 효율적인 초기화 단계**를 제안한다. 이는 비정지 초기 state에서도 정확한 초기 states, 중력 벡터, 일관된 초기 local map을 제공한다. 또한 초기화 모듈은 초기화 성공 여부를 판단하고, 중력 벡터를 월드 프레임의 z 축에 재정렬할 수 있다.

초기화는 첫 **N개의 LiDAR scan**과 대응되는 IMU measurement들을 사용하여 수행된다.

1. 구체적으로, 먼저 **[Fast-lio]의 LIO 방법**을 실시간으로 실행하여 N개의 scan에 대한 **대략적 state**와 중력 벡터를 추정한다.
2. 이렇게 추정된 state와 중력 벡터는 이후의 **coarse-to-fine voxelization**과 **BA optimization**(V-A절)을 위한 초기 값(initial value)으로 사용된다.
3. 그 결과로 얻은 최적화된 state와 중력 벡터는 초기화 성공 여부를 판단하는 데 사용된다(V-B절).

4. 성공하면 초기화가 완료되고(V-C절), 실패하면 다음 N개의 scan을 수집하여 성공할 때까지 위 과정을 반복한다.

A. Coarse-to-Fine Voxelization and BA Optimization

- 대략적 initial states(예: LIO로 추정한)를 바탕으로, **coarse-to-fine voxelization and BA optimization**는 LiDAR-Inertial BA(IV절)를 이용해 state와 대응되는 map을 정제(refine)한다. 부정확한 initial states는 정합된 포인트클라우드 맵에 큰 불일치를 일으켜, adaptive voxelization 과정에서 잘못된 평면 연관성(false positive or false negative plane associations)을 유발할 수 있다(III-C절 참고).이러한 잘못된 평면 연관성은 BA optimization error를 발생시킨다.

  이를 극복하기 위해, 본 연구에서는 adaptive voxelization 과정(섹션 III-C)에서 plane feature를 결정하는 기준을 점진적으로 강화하는 **coarse-to-fine voxelization and BA optimization**를 제안한다.

  1. 우선 느슨한(loose) 평면 판별 기준으로, 큰 initial state error가 있더라도 가능한 한 많은 plane feature를 회수(recall)한다.
  2. 회수된 plane feature를 이용해 LiDAR-Inertial BA를 수행해 state를 정제(refine)한다.
  3. 정제된 state는 각 scan의 motion distortion 보정에 사용되고, 이전보다 더 **엄격(tight)한 평면 판별 기준을 가지는 다음 라운드의 adaptive voxel map**을 구축하는 데 사용된다.
  4. 이렇게 **voxelization과 BA optimization**은 여러 라운드 반복하며, 각 라운드마다 기준을 강화한다. 직전 라운드에서 현재 라운드 **cost value 감소량**이 특정 임계값 이하가 되면 수렴(convergence)으로 간주한다.

B. Criteria for Initialization Success

- 초기화 성공 여부는 세 가지 기준으로 판단한다:
  - ① coarse-to-fine process가 **특정 라운드 수** 이내에 수렴할 것
  - ② 최적화된 중력 벡터의 크기가 **9.8 m/s²**에 근접할 것
  - ③ 최적화된 state로 생성된 초기 voxel map에 **선형 독립적**인 세 방향의 평면 제약이 존재할 것.
- 이러한 degeneration 평가는 V-B Appendix의 degeneration 판별에 자세히 설명되어 있다.

만약 마지막 BA optimization에서 degeneration이 발생한다면, 이는 초기화에 포함된 states가 충분한 plane features으로 제대로 제약되지 않아 estimation errors가 발생할 수 있음을 의미한다.

C. Completion of Initialization

- 초기화에 성공하면, **N개 스캔에서** distortion이 보상된 point clouds을 포함하는 정확한 중력 벡터와 states를 획득한다.이후 회전 행렬 R_g^z을 사용해 중력 벡터가 월드 프레임의 z축과 맞도록 재정렬하고, 모든 states 역시 이 회전 행렬에 의해 동일하게 회전된다.
- 중력 벡터는 이후 모듈들에서 **고정(fixed)**값으로 사용된다. 최적화된 state로부터 구축된 adaptive voxel map은 **Odometry 모듈**의 초기 local map으로 사용된다.
- **초기화 모듈은 각 시작(또는 재시작)마다 한 번만 수행되므로, 초기화 모듈 이후에는 LiDAR scans 및 IMU measurements가 초기화 없이 바로 Odometry에 전달된다.**

▼ Odometry

Voxel-SLAM에서 오도메트리 모듈은 **scan-to-map registration**을 통해 **short-term data association**을 활용한다. 오도메트리의 목적은 planning·control 등 다른 작업에 필요할 정도로 **지연을 최소화**하여 현재 state를 estimation하는 것이다. 이렇게 추정된 state는 후속 **Local Mapping** 모듈에 비교적 정확한 초기값을 제공한다.

A. State Prediction and Update

local adaptive voxel map 상에서 state estimation을 수행하기 위해, **우리는 scan-to-map correspondence에는 [Voxel map]의 방법을, LiDAR와 IMU measurement을 tightly-coupled 하기 위해 [Fast-lio]의 Extended Kalman Filter(EKF)를 사용**한다.

1. 구체적으로, LiDAR scan과 IMU measurement이 동기화(Sync)되면, 오도메트리는 **IMU propagation**을 통해 state를 predict하고 동시에 LiDAR scan의 **motion distortion**을 보정한다.왜곡 보정된 scan의 포인트들은 공간적으로 해상도 L_d로 **다운샘플링**되고, 사전에 알려진 LiDAR-IMU extrinsic 파라미터를 이용해 **LiDAR body frame에서 IMU body frame으로 변환**된다.
2. scan의 각 포인트는 **[Voxel map]**의 방법으로 공분산 행렬을 계산한 뒤, 월드 프레임으로 projection되어 voxel map에서 **최대 확률을 갖는 평면**과 매칭된

다.

3. **Point-to-plane** 거리는 시스템 measurements으로 사용되며, 이는 **error-state(iterative) Kalman filter** [Fast-lio] 내부에서 IMU predicted state를 업데이트한다.
4. 이렇게 추정된 state와 LiDAR scan은 다른 모듈에서 추가 처리를 위해 퍼블리시된다.

B. Detection of System Divergence

로봇이 충분한 기하학적 features가 부족한 **degenerate environment(열악한 환경)**를 만나면, state estimation이 제대로 이뤄지지 않는다(ill-posed). 이는 SLAM 시스템이 **발산(diverge)하여 실패**로 이어질 수 있으므로, 적극적으로 모니터링하고 처리해야 한다.
Voxel-SLAM에서는
각 scan registration 이후 시스템의 잠재적 diverge를 감지한다. 오도메트리 모듈에서 state estimation이 끝난 뒤, scan-to-map 정합에 사용된 평면들을 모아 부록 B(Appendix B)의 방법으로 퇴화(degeneration)를 평가한다.

- 단일 scan에서 가끔 발생하는 degeneration은, 짧은 시간 동안 IMU prediction이 비교적 정확하므로 diverge로 간주되지 않는다.
- 그러나 **여러 연속 scan**에 걸쳐 지속적으로 발생하는 degeneration은 diverge로 판단한다.
    - 이런 경우, 현재 데이터 세션을 저장하고, 다음 세션의 포인트가 현 세션과 겹치지 않도록 충분히 떨어진 새 원점에서 **시스템을 재시작**한다.
    - 재시작은 또한 다음 세션을 위한 **새로운 Initialization 모듈**을 실행하여 initial state와 local map을 다시 획득한다.

▼ Local Mapping

Local mapping은 mid-term data association(중기 데이터 연관성)을 활용하여, 슬라이딩 윈도우 내의 모든 최근 states들과 local voxel map을 정제(refine)한다. Odometry 모듈과 비교했을 때, Local mapping 모듈은 LiDAR-inertial BA가 LiDAR 및 IMU의 더 긴 기간의 measurement를 활용하여 state를 estimation하기 때문에 정확도와 강건성이 더 높다.

A. Keyframe Loading and Voxelization

Local mapping 모듈이 Odometry 모듈로부터 추정된 **estimated state와 함께 현재 LiDAR scan, IMU measurements**을 받으면, scan의 각 포인트들은 월드

프레임에서의 위치에 따라 대응되는 voxel에 분배된다.

동시에 IMU measurements는 [26]에 따라 **pre-integration**되어 IMU preintegration factor를 생성한다.

Local mapping은 현재 LiDAR 위치 주변 반경 **L_m** 내 local map만 유지하기 때문에, 이 voxel map은 현재 혹은 이전 세션에서 존재했던 장소를 재방문할 경우 long-term data association(장기적 데이터 연관성)을 보장할 수 없다.

장기적 데이터 연관성을 확보하기 위해, 우리는 **시간적으로 멀리 떨어져 있으나(예: ≥ 1분, 최근 keyframe들을 피하기 위해) 공간적으로는 가까운(예: ≤ 10 m)** keyframe들을 동적으로 로드한다 — 이 기법은 [LTA-om]에서 효과가 입증되었다.

이렇게 불러온 keyframe들의 포인트들은 대응되는 voxel에 분배되어, 슬라이딩 윈도우 BA optimization을 위한 **고정된 과거 맵 제약(fixed historical map constraints)에 사용**된다.

현재 scan이나 동적으로 로드된 keyframe의 포인트를 voxel에 배치할 때, 우리는 **새롭게 포인트가 추가된 voxel**을 기록(tracking)한다.

이러한 voxel을 열거(enumerate)하여, 그 안의 모든 포인트(현재 scan·keyframe·기존 포인트 포함)를 **Adaptive voxelization(III-C 절)** 과정에 통과시켜 각 leaf node가 유효한 plane을 갖도록 한다.

각 leaf node에서 모든 포인트는 **point cluster** [BALM 참고]라는 압축 구조로 인코딩되며, 슬라이딩 윈도우 내 서로 다른 스캔은 서로 다른 point cluster를 갖는다.

leaf node에 남은 포인트는 **"fix" 포인트**라 불리며, 슬라이딩 윈도우 최적화를 위한 맵 제약조건(map constraints)로 작용한다(이 "fix" 포인트들은 동적 keyframe 또는 이전에 marginalize된 scan에서 올 수 있다; VII-C(Map Marginalization) 절 참조).

이러한 "fix" 포인트들은 하나의 point cluster로 인코딩되며, 슬라이딩 윈도우 내 각 scan의 point cluster와 함께 **개별 포인트를 열거하지 않고도** BA optimization에 직접 사용된다.

B. Sliding Window BA Optimization

point cluster 연관성과 IMU preintegration을 확보한 뒤, **Sliding-window BA 최적화**는 Odometry 모듈의 LiDAR-Inertial BA를 이용해 슬라이딩 윈도우 내 모든 state를 정제한다.

Sliding-window BA와 Odometry 모듈의 BA의 차이점

1. **각 voxel의 "fix" 포인트들로부터 제약사항을 통합해** 월드 프레임에서 map constraints로 제공되어 estimation 정확성을 보장한다.
2. 초기화 단계에서 이미 중력 벡터를 추정했으므로 **중력 벡터를 고정(fixed)** 한다.

그림 3은 Sliding-window LiDAR-Inertial BA optimization의 **factor graph** 표현이다.

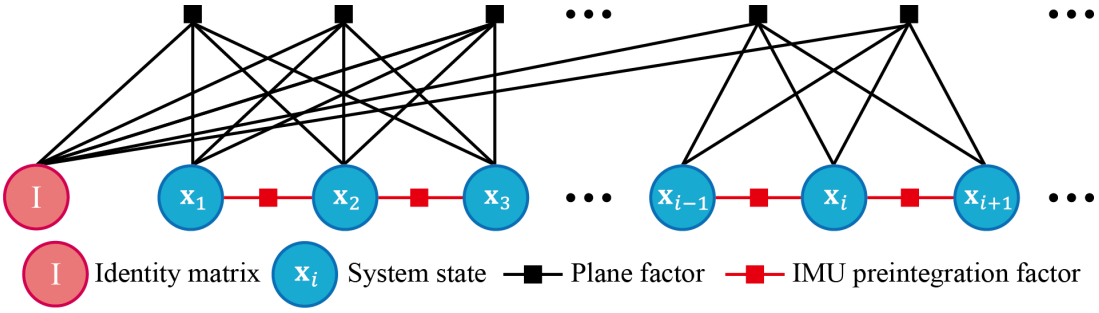

그림 3. local mapping에 사용된 LiDAR-inertial BA의 factor graph 표현. **"identity matrix"은 슬라이딩 윈도우 최적화 범위를 벗어나 거나 월드 프레임에 표현된 "fix" 지점(point cluster 형태로 표현됨)로부터의 map constraints을 나타낸다.**

C. Map Marginalization

BA optimization 이후, local voxel map의 point cluster와 plane 파라미터(평면 중심·법선 벡터)는 정제된 state에 의해 업데이트된다.

업데이트된 plane은 다음 번 odometry 모듈의 **scan-to-map correspondence**에 사용된다.

또한 슬라이딩 윈도우에서 **가장 오래된 scan**은 **marginalize**된다. 즉, 가장 오래된 scan은 윈도우에서 제거되고, 각 leaf node의 해당 scan의 포인트들은 map constraints으로써 "fix" point cluster로 병합된다.

Marginalization이 끝나면, 가장 오래된 scan과 해당 scan의 pose는 **Loop Closure** 모듈로 전달되어 추가 처리를 받는다.

또한, 슬라이딩 윈도우 내 state에 대응하는 포인트가 leaf node에 없는 voxel들이나 현재 pose로부터의 거리가 local map 크기 L_m보다 큰 voxel들은 메모리 사용량을 줄이기 위해 local map에서 제거된다.

▼ Loop Closure

Loop closure는 본 프레임워크 안에서 두 가지 중요한 역할을 수행한다.

1. 첫째, 장기 데이터 연관성(long-term data association)을 활용하여 이전에 방문한 위치를 감지함으로써 drift를 효과적으로 완화하고, pose graph

optimization(PGO)를 통해 누적 오차를 보정한다.

2. 둘째, loop closure는 동일한 월드 프레임에서 현재 세션을 이전 세션들과 매칭하여 **multi-map data association**을 달성한다.이 프로세스를 통해 여러 매핑 세션에서 수집된 데이터를 통합하고 정렬하여 통합된(unified) 환경 map을 효율적으로 생성한다.

A. Keyframe Generation and Loop Detection

Local Mapping의 슬라이딩 윈도우에서 마지널라이즈된(marginalized) scan은 Loop Closure 모듈에 의해 pose graph에 추가된다.

N개의 연속된 마지널라이즈 scan은 하나의 frame으로 누적된다. 현재 frame과 최신 keyframe 사이의 이동 거리 및 회전 각이 사전에 정한 임계값(예: 0.5 m, 5 deg)보다 크면 해당 frame이 **keyframe**으로 선택된다.

이렇게 선택된 keyframe은 **BTC descriptor**를 추출하고, 현재 및 이전 세션의 후보 loop keyframe들을 매칭하는 데 사용된다. 이는 [BTC 논문]에 자세히 설명되어 있다. 동시에 이 keyframe은 추가 처리를 위해 Global Mapping 스레드에도 전달된다.

Loop closure 후보가 식별되면, 오탐(false-positive)을 방지하기 위해 그 진위를 검증하는 것이 필수적이다.

BTC [18]에 내장된 기하학적(geometric) 검증 외에도, 우리는 loop 검출의 신뢰성을 높이기 위해 두 가지 추가 기준을 적용한다.

(1) 현재 keyframe과 검출된 후보 사이의 **point-to-plane alignment**는 **Appendix B의 기준에 따라 세 개의 선형 독립 방향에서 평면 제약을 포함**해야 한다.

(2) **drifting distance**와 **traveling distance**의 비율**(이동 거리에 대한 드리프트 거리의 비율)**이 충분히 작아야 한다(예: ≤ 1%).

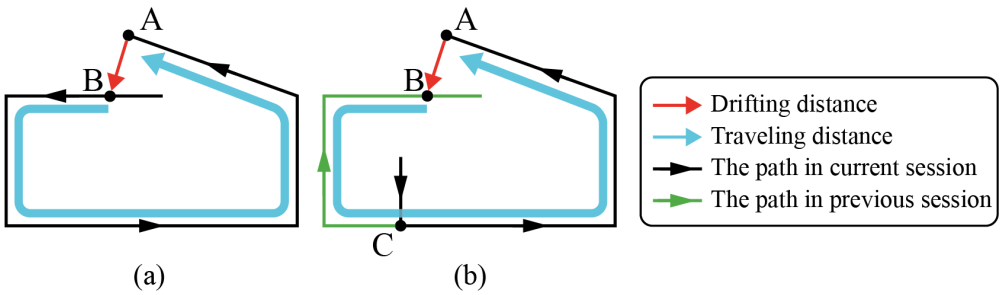

그림 4. 드리프트 및 이동 거리(Drifting and traveling distances).
(a) 후보 loop keyframe은 현재 세션에 있다. 이동 거리는 후보 loop keyframe B에서 현재 keyframe A까지 현재 세션을 따라 누적된 거리이다.
(b) 후보 loop keyframe은 이전 세션에 있다. 이동 거리는 후보 loop keyframe B에서 이전 loop keyframe C까지 이전 세션을 따라 누적된 거리와 그 후 현재 세션을 따라 C에서 현재 keyframe A까지 누적된 거리의 합이다.
두 경우 모두 드리프트 거리는 현재 keyframe A와 후보 loop keyframe B 사이의 상대 거리이며, 드리프트 거리는 loop detection 방법 BTC[18]에서 계산된다.

- 이 두 번째 기준은 후보 loop keyframe이 현재 세션에 있거나(그림. 4(a) 참조) 이전 세션에서 두 번째(혹은 그 이상)로 탐지될 때(그림. 4(b) 참조) 적용된다.
- 이 기준은 성능이 우수한 SLAM 시스템에서 일정한 주행 거리를 이동한 후 발생하는 drift가 대체로 일정 범위 이내라는 관찰에 기반한다.
- 만약 이 비율이 너무 크면, 해당 loop 탐지는 오탐일 가능성이 높다.

또한 후보 loop keyframe이 처음으로 **이전 세션** 내에 있는 경우에는 두 번째 기준**(이동 거리에 대한 드리프트 거리의 비율)**이 적용되지 않으며, **첫 번째 기준(평면 제약)과 BTC [18] 내부 기하 검증**을 더욱 엄격히 해야 한다. 이는 서로 다른 세션 간 첫 loop 검출이 잘못될 경우 시스템 후속 동작에 치명적인 영향을 줄 수 있기 때문이다.

B. Pose Graph Optimization and Map Rebuilding

Loop 검출이 성공적으로 통과했을 때:

- (Case 1) 만약 검출된 loop keyframe이 처음으로 **이전 세션에** 있는 경우
  - **현재 세션의 모든 pose를 이전 세션의 월드 프레임에 정렬(alignment)하고 두 세션의 pose graph를 연결(concatenate)**한다. 즉, 현재 세션이 이전 세션에 연결된다.
- (Case 2) 검출된 loop keyframe이 두 번째(이상) 있는(이미 1번으로 연결된) 이전 세션에 있거나, 현재 세션 내에 있는 경우

- pose graph 내부에 loop가 형성되며, 이때 **GTSAM** [48]을 이용한 PGO가 트리거된다.

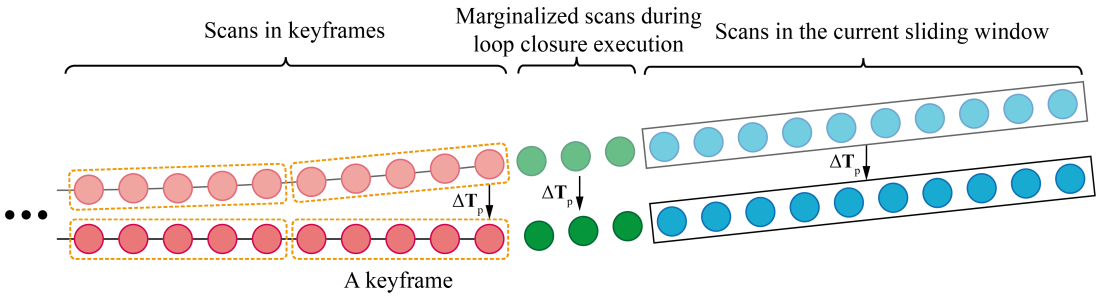

그림 5. PGO 전후 scan.

빨간색 노드는 pose graph에 추가된 scan이다.

녹색 노드는 loop closure 실행 중 local mapping에서 marginalize된 scan으로, 아직 pose graph에 추가되지 않았다.

파란색 노드는 local mapping의 현재 슬라이딩 윈도우에 있는 scan이다.
상단과 하단의 노드는 각각 PGO 전후 scan이다.

ΔT_p는 pose graph의 마지막 scan에 대한 pose 보정이며, pose graph 이후의 후속 scan을 보정하는 데 사용된다.

PGO를 통해 scan pose가 업데이트된 후, keyframe과 submap의 pose 역시 해당 업데이트된 scan pose에 맞춰 동기화된다(그림. 1의 데이터 피라미드 및 그림. 5의 빨간 노드 참고).

더 나아가 pose graph에 포함되지 않은 scan들의 pose도 시스템 일관성을 위해 update해야 한다.

- 이러한 스캔은:
  - (1) Loop Closure 수행 중에 마지널라이즈된 scan들(Fig. 5의 초록 노드)과
  - (2) 마지막 scan이 pose graph에 추가된 이후 앞으로 이동한 현재 슬라이딩 윈도우의 모든 scan(그림. 5의 파란 노드)으로 구성된다.

이 두 종류 scan은 모두 ΔT_p 변환에 의해 pose가 보정되는데, **ΔT_p는 그림. 5에 표시된 pose graph 내 마지막 scan의 pose보정값**이다.

pose가 update된 두 종류의 scan과 pose graph의 최신 5개 keyframe의 모든 scan은 새로운 adaptive voxel map을 구축하는 데 사용된다.

- (1) 새로운 voxel map의 각 leaf 노드에서는 keyframe과 마지널라이즈 scan의 포인트가 **"fix" 포인트**로서 지도 제약(map constraint)을 제공하므로 하나의 **point cluster**로 인코딩된다.
- (2) 반면, 현재 슬라이딩 윈도우에 속한 scan 포인트는 Section VII-A에서 설명한 대로 각각 별도의 point cluster로 인코딩된다.

마지막으로, **이 새로운 voxel map은 기존 voxel map을 대체하여 향후 Odometry 및 Local Mapping 단계에서 사용된다(즉, voxel map이 최적화되어 업데이트 되었단 소리).** 계산 자원을 절약하기 위해, **PGO and map rebuilding은** Section VIII-A에서 정의한 drifting distance가 미리 정한 임계값(예: 0.1 m)보다 클 때만 수행된다.

▼ Global Mapping

다중 세션 SLAM 환경에서는, Loop Closure 모듈로부터 생성된 포인트 클라우드 맵이 **PGO에서 pose constraints만 고려**하기 때문에 일관성이 부족할 수 있다. 이는 직접적인 맵 일관성(**map consistency**) constraints를 반영하지 못하기 때문이다.

이 문제를 해결하기 위해, 우리는 **계층형 global bundle adjustment (HBA)** 방법을 사용한 **Global Mapping** 모듈을 도입하여 **Loop Closure 모듈로부터 생성된 pose와 map을 추가로 정제**한다.

그림 1의 data pyramid를 기반으로, Global Mapping 모듈은 다음 세 가지 절차를 포함한다.

- (Process 1) Loop Closure 모듈에서 keyframe을 수신하면, Global Mapping 모듈은 실시간으로 keyframe 윈도우에 대해 BA를 수행한다.
  - 윈도우의 크기는 10개의 keyframe이고, stride는 5이다.
  - 각 슬라이딩 윈도우 BA에서 첫 번째 keyframe의 pose는 PGO 결과에 고정되고, 나머지 9개의 keyframe pose는 최적화 과정에서 동시에 최적화된다.
  - 이후 윈도우 내 10개 keyframe은 하나의 **submap**으로 병합된다.
- (Process 2) 현재 세션이 종료되면(예: 데이터 수집 종료 또는 시스템 재시작), submap들을 대상으로 Global BA를 수행한다.
  - PGO에서 반환된 scan pose가 충분한 global map 일관성을 제공하지 못할 수 있다는 점을 고려하여, Section V-A에서 제안한 **coarse-to-fine voxelization and BA optimization** 기법을 global submap BA에 적용한다.

    - 여기서 IMU pre-integration factors와 중력 최적화는 제거된다.

keyframe → submap → global map으로 이어지는 이 BA 최적화는 [HBA]에서 **bottom-up BA**라고 하며, keyframe이나 scan에 직접 global BA를 수행하는 것보다 최적화 차원을 크게 줄여준다.

- (Process 3) PGO를 결합한 [HBA]의 **top-down optimization**을 적용해 전역적인 일관성을 향상시킨다.

▼ Experimental Results

우리는 시스템의 정확도와 다재다능성을 검증하기 위해 다음 5가지 실험을 수행한다.

1. 초기화(Initialization)의 견고성, 정확도, 시간 소모 평가
2. 단일 세션 SLAM에서 다른 odometry 및 SLAM 방법과의 벤치마크 비교
3. 다중 세션 SLAM 실험
4. 연산 자원이 제한된 컴퓨터에서 온라인 재현지화(relocalization)를 포함한 다중 세션 SLAM
5. 위 2)–4) 실험의 연산 시간 분석

우리는 특성이 뚜렷이 다른 세 개의 공개 데이터셋—**Hilti**[49, 50], **MARS-LVIG**[51], **UrbanNav**[52]—과 두 개의 개인 데이터셋을 사용했으며, 총 32개 시퀀스를 포함한다. 각 시퀀스의 시간·거리 정보는 Appendix C의 Table VIII에 제시되어 있다.

1. **Hilti**는 실·내외 구조물 공사 현장에서 수집된 SLAM 데이터셋으로, Hesai XT-32 LiDAR와 Bosch BMI085 IMU(400 Hz)로 취득된 handheld 시퀀스를 사용하며, 웹사이트에서 ATE(Absolute Trajectory Error)를 제공한다.
2. **MARS-LVIG**는 고도 100 m 상공에서 downward-looking 방식으로 Livox Avia LiDAR(내장 IMU BMI088, 200 Hz)로 수집된 UAV 데이터셋이다.
3. **UrbanNav**는 Xsens-MTI-30 IMU(400 Hz)와 HDL-32E LiDAR를 장착한 도시 환경 로봇카 데이터셋이다.
4. 개인 데이터셋은 Livox Avia LiDAR와 내장 IMU를 장착한 경량 handheld 장치로 수집되었으며 Fig. 13에 장치를 보여준다.

모든 시퀀스에서 LiDAR는 10 Hz로 동작한다.

실험 환경 & 공통 파라미터

1. 앞의 세 실험(Initialization, 단일·다중 세션 SLAM)은 Intel i7-10750H CPU(3.5 GHz), 메모리 32 GB 노트북에서 수행되었다.
2. 네 번째 실험(온라인 재현지화)은 Intel i3-N305 CPU(3.0 GHz), 메모리 16 GB 온보드 컴퓨터에서 수행되었다.
3. Root voxel 크기와 스캔 다운샘플 해상도는 환경별로 다음과 같이 설정했다: 실내 $L_r = 1m, L_d = 0.1m$, 실외 $L_r = 2m, L_d = 0.25m$, 고고도 하향 시야 $L_r = 4m, L_d = 0.5m$.
4. 로컬 voxel 맵 반경은 $L_m = 1,000m$, 최대 계층은 $l_max = 3$, 한 평면의 최소 포인트 수는 $N_{min} = 5$, 평면 판별 기준은 $\lambda_3/\lambda_2 < 1/16$이다$(\lambda_1 \geq \lambda_2 \geq \lambda_3)$.
5. 초기화, LiDAR-Inertial BA, Keyframe BA의 슬라이딩 윈도 크기는 $N = 10$으로 모든 데이터셋에서 동일하게 설정했다.
6. 추가 실험 세부 정보는 비디오에서 확인할 수 있다.

A. Initialization

1) 정성적(qualitative) 분석

초기화 모듈의 강건성을 검증하기 위해, 구조물이 없는 숲 환경에서 수집한 개인 시퀀스 *"private1"*에 대해 테스트를 수행하였다.

해당 시퀀스는 초기 각속도 83.1 deg/s, 선형 가속도 6.7 m/s²(중력 제외)와 같이 빠른 회전·진동을 포함한다.

- 그럼에도 Voxel-SLAM은 성공적으로 초기화를 수행해 그림. 6(a)와 같이 실제 움직임을 반영한 일관된 지도와 궤적을 생성했다.
- 반면 FAST-LIO2는 강건한 초기화 모듈이 없어 시작 단계에서 발산하였다
    - 이는 짧은 IMU 구간 평균 가속을 9.8 m/s²로 스케일해 중력을 추정하는데, 초기 가속이 6.7 m/s²로 커서 실제 중력 방향과 크게 어긋났기 때문이다.

초기화 모듈의 효과를 추가로 검증하기 위해, 시퀀스의 1/7부터 6/7 지점까지 다양한 시점에서 시스템을 시작했으며 그림. 6(c)–(h)에 결과를 제시한다.

- 이는 다양한 초기 조건에서도 초기화 모듈이 높은 강건성을 보임을 입증한다.

2) Quantitative analysis

다음으로, 우리는 **UrbanNav**[52]의 "urban1" 시퀀스를 사용하여 초기화 모듈에 대한 정량적 분석을 수행한다.

- 이 데이터셋은 고속도로 주행 환경에서 수집되었으며 1 Hz 주기로 **ground-truth** 포즈와 속도를 제공한다.

초기화 모듈을 시퀀스 내 서로 다른 시작 시점에서 반복 실행하였고, 각 경우마다 초기 속도 및 중력 벡터—초기화 성능에 중요한 두 파라미터—를 평가하였다. 평가를 위해 "urban1"은 IMU 바디 프레임에서의 ground-truth 속도를 제공한다.

초기화가 완료된 뒤, 우리는 추정된 IMU 상태의 속도를 IMU 바디 프레임으로 변환하여 ground-truth와 직접 비교한다.

ground-truth 중력 벡터는 데이터셋에 직접 제공되지 않으므로, 시퀀스 시작 후 첫 1초 동안의 IMU 평균 가속도를 중력으로 간주하였다.

이 구간 동안 IMU가 정지 상태임은 ground-truth 속도와 카메라 영상을 통해 확인하였다(IMU와 카메라는 리지드하게 결합됨).

초기화에서 계산된 상대 스캔 포즈와 바이어스는 평가 대상이 아니다; ground-truth 상대 스캔 포즈는 1 Hz 해상도로는 얻을 수 없고, 실제 환경에서 IMU 바이어스의 ground-truth도 알 수 없기 때문이다.

시퀀스 내 서로 다른 시작 시점뿐 아니라, 슬라이딩 윈도 크기(5, 10, 20, 40)가 초기화 성능에 미치는 영향도 탐색하였다.

- 그림. 7은 혼합된 구조·비구조(Scene) 환경에서 실행된 초기화 실험 중 하나의 포인트클라우드 맵을 보여준다. 이 경우 초기 속도는 9.6 m/s였다.
- 그림. 7(b, d, f)는 LIO 결과(map)로, 이는 초기화 과정에서 BA 최적화의 입력이다.
- 그림. 7(c, e, g)는 coarse-to-fine voxelization과 BA 최적화 이후 정제된 포인트클라우드를 보여준다.

나무 줄기와 벽면 포인트가 훨씬 일관되게 정렬됨으로써 초기화의 효과를 확인할 수 있다.

ground-truth와 추정된 초기 속도·중력 벡터 비교 결과는 그림. 8과 Table I에 제시된다.

- 모든 윈도 크기(5–40)에서, 초기 속도는 시작 시점과 매우 높은 초기 속도(> 10 m/s)에도 불구하고 잘 추정되었다.
- 중력 벡터 추정 오차는 윈도 크기에 따라 감소한다; 더 큰 윈도는 더 긴 IMU 데이터를 사용하여 정보량을 늘리기 때문이다.
- 윈도 크기가 10을 초과해도 속도·중력 오차가 눈에 띄게 줄지 않는 반면 계산 시간은 크게 증가하므로, 우리는 Voxel-SLAM에서 N = 10을 채택하였다.

B. Single-Session SLAM

이 절에서는 제안 시스템 **Voxel-SLAM**을 최신 오픈소스 LiDAR(-Inertial) odometry 및 SLAM과 비교한다.

- 비교 대상 odometry는 **LINS**[27], **FAST-LIO2**[13], **Faster-LIO**[14], **Point-LIO**[15]이고, SLAM은 루프 클로저를 포함한 **LeGO-LOAM**[21], **LiLi-OM**[24], **LIO-SAM**[25], **LTA-OM**[10]이다.
- 모든 비교 방법은 별도 언급이 없는 한 기본 파라미터를 사용하였다.
- Odometry 정확도 평가를 위해 SLAM 방법들의 루프 클로저 기능은 비활성화하였다.
- 아울러 우리는 모듈별 ablation을 통해 "Our(Odom)", "Our(Odom + LM)", "Our(Odom + LM + LC)", "Our(Full)" 네 구성으로 성능을 제시한다.

1) Hilti Handheld Dataset

- Table II는 모든 방법의 odometry 및 SLAM ATE를 나타낸다.
- Hilti 데이터셋의 주요 난점은 좁은 계단 통로이며(sequence 03, 05, 07, 13), LiDAR odometry에는 매우 도전적이다.
  - LeGO-LOAM, LiLi-OM, LINS, LIO-SAM은 이러한 시퀀스에서 실패하였다.
  - FAST-LIO2 계열(LIO) 방법은 더 견고하였으나 여전히 큰 자세 오차를 보였다.
  - Voxel-SLAM의 odometry는 adaptive voxel map 덕분에 작은 오차를 달성하였다.
    - Local Mapping을 추가하면 중기 연관성 덕분에 정확도가 더욱 향상되어 모든 시퀀스에서 최고 성능을 기록한다.

        - 루프 클로저가 활성화된 전체 SLAM 결과에서도, Voxel-SLAM은 Global Mapping 없이도 최고 정확도를 달성하며, Global Mapping까지 포함하면 모든 시퀀스에서 오차가 추가로 줄어든다.
        - 이는 루프가 없더라도 Global Mapping이 장기 연관성을 활용해 정확도를 높일 수 있음을 시사한다.

2) MARS-LVIG Aerial Dataset

- 이 데이터셋은 고도 100 m, downward-looking 시야를 가지며 solid-state LiDAR를 사용한다.
    - LeGO-LOAM과 LINS는 이 LiDAR 유형을 지원하지 않아 실행 불가였고, LIO-SAM은 개발자 가이드에 따라 수정 후 실행하였다.
    - Table III 결과에서 Voxel-SLAM은 대부분 시퀀스에서 가장 낮은 ATE를 기록한다.
        - 특히 "mars4" 그룹(고속 비행, 부분적 퇴화 구간)에서 FAST-LIO2 등은 지형 정상(hilltop)에서 포인트클라우드가 두 층으로 분리되는 등 불일치가 발생하였으나, Voxel-SLAM은 Local Mapping과 Global Mapping 덕분에 이를 완화하였다(그림. 10).

3) UrbanNav Robot-Car Dataset

- UrbanNav는 최대 13 m/s 속도, 56분 이상 수집 등 Hilti보다 훨씬 큰 규모와 동적 객체가 많다.
    - Table IV에서 Voxel-SLAM은 odometry와 SLAM 모두 전 시퀀스에서 최고 정확도를 달성하였다.

C. Multi-Session SLAM

- Hilti 09–13 시퀀스는 동일 공사 현장에서 취득되어 동일 월드 프레임 ground-truth를 공유한다.
    - 우리는 "hilti13→12→11→10→09" 순서로 Voxel-SLAM에 입력하여 다중 세션 평가를 수행하였다.
        - 각 시퀀스가 끝나면 해당 세션을 메모리에 저장하여 후속 세션에서 루프를 탐색할 수 있도록 했다.
        - 그림. 11(a)는 세션 간 루프 검출로 형성된 연결 그래프를 보여주며, 그림. 11(b)는 PGO와 Global Mapping 이후 정렬된 궤적을 나타낸다.

    - 일부 좁은 복도에서는 루프가 누락되었으나(Global Mapping 전), Global Mapping이 이를 보정하여 전체 지도 일관성을 개선하였다(그림. 12).
    - Table V는 PGO만 적용한 경우와 Global Mapping까지 적용한 경우의 ATE를 비교하며, 모든 세션과 전체(multisession) 오차 모두가 감소했음을 보여준다.

D. Relocalization

다음 실험은 퇴화 이후 재초기화 및 재현지화 능력을 평가한다.

Intel i3-N305(연산 제약) 온보드 컴퓨터에서 handheld 시퀀스 "private2"를 실시간으로 처리하였다.

- 그림. 13–15는 엘리베이터 진입 시 divergence 탐지 및 세션 분할, 엘리베이터 탈출 후 재초기화, 스타트 지점 재방문 시 루프 클로저를 통한 세션 재현지화 과정을 보여준다.
- 재현지화 이후 Local Mapping·Loop Closure가 장기·다세션 연관성을 유지하여 전역 지도를 일관되게 확장한다(그림. 15).

E. Computation Time

모든 실험 시퀀스에서 평균 연산 시간과 메모리 사용량을 평가하였다(Table VI).

LiDAR 스캔은 10 Hz이므로, Odometry·Local Mapping은 자연스럽게 10 Hz로 실행된다.

- Loop Closure와 Keyframe BA는 이벤트 기반이므로 전체 소요 시간을 스캔 수로 나누어 평균 per-scan 시간을 계산하였다.
    - 세 개의 스레드(1. Odometry + Local Mapping, 2. Loop Closure, 3. Keyframe BA)는 모두 스캔 간격 $t_s = 0.1s$보다 짧은 평균 시간을 보여 실시간성이 입증된다.
- Initialization과 Global Mapping은 세션당 한 번 실행되므로, 실행 횟수당 평균 시간을 제시하였다.
    - 가장 긴 "urban3" 시퀀스조차 Global Mapping 시간이 79.7초로, 56분 데이터 수집 시간 대비 2.4%에 불과하다.
- 메모리 사용량 또한 노트북·온보드 환경 모두 물리 메모리 이하로 안정적으로 유지되었다.

▼ Conclusion And Future Works

본 논문에서는 **Voxel-SLAM**이라는 완전하고 고정밀한 범용 LiDAR-Inertial SLAM 시스템을 제안하였다. 이 시스템은 initialization, odometry, local mapping, loop closure, and global mapping 모듈을 모두 포함하며, 전 단계에서 동일한 adaptive voxel map 구조를 사용한다.

- Initialization 모듈은 매우 빠르고 강건하게 작동하여 이후 모듈들이 사용할 수 있는 정확한 상태와 일관된 지도를 제공한다.
- Odometry 모듈은 현재 상태를 신속히 추정하면서 잠재적인 시스템 divergence를 탐지한다.
- Local Mapping 모듈은 효율적인 tightly-coupled LiDAR-Inertial Bundle Adjustment를 활용해 상태와 지도를 동시에 정제하여 정확도와 강건성을 향상시킨다.
- Loop Closure 모듈은 여러 세션에서 재방문 장소를 탐지할 수 있다.
- Global Mapping 모듈은 Data Pyramid를 기반으로 효율적이고 정확하게 설계되었다.

이들 모듈은 단기, 중기, 장기, 다중 지도 등 네 가지 데이터 연관성을 모두 활용한다.

향후 시스템은:

- 이미지 측정을 융합하여 LiDAR degeneration 환경에서의 강건성을 보강하고, 포인트클라우드에 색상 정보를 부여하며, 장소 인식 성능을 향상시킬 수 있을 것이다.
- 현 시스템은 온보드 컴퓨터에서도 실시간으로 동작할 만큼 충분히 효율적이지만, 특히 Global Mapping 단계에서 GPU 병렬화를 도입하면 전체 시스템 효율을 더욱 향상시킬 수 있다.
    - Adaptive voxel map 프레임워크는 이러한 병렬 처리에 매우 적합하다.

- 논문 내용 정리
- 오픈소스 코드 분석(notion 내 링크 추가 예정)