

# R<sup>3</sup>LIVE++: A Robust, Real-time, Radiance reconstruction package with a tightly-coupled LiDAR-Inertial-Visual state Estimator

Jiarong Lin and Fu Zhang

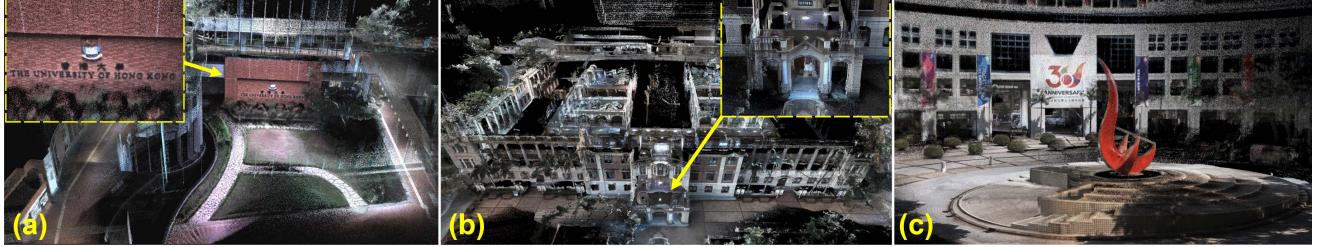


Fig. 1 – The radiance map of HKU (a and b) and HKUST campuses (c) reconstructed by R<sup>3</sup>LIVE++ in real-time (see our accompanying video on YouTube; [youtu.be/qXrnIfn-7yA](https://youtu.be/qXrnIfn-7yA)).

**Abstract**—This work proposed a LiDAR-inertial-visual fusion framework termed R<sup>3</sup>LIVE++ to achieve robust and accurate state estimation while simultaneously reconstructing the radiance map on the fly. R<sup>3</sup>LIVE++ consists of a LiDAR-inertial odometry (LIO) and a visual-inertial odometry (VIO), both running in real-time. The LIO subsystem utilizes the measurements from a LiDAR for reconstructing the geometric structure, while the VIO subsystem simultaneously recovers the radiance information of the geometric structure from the input images. R<sup>3</sup>LIVE++ is developed based on R<sup>3</sup>LIVE and further improves the accuracy in localization and mapping by accounting for the camera photometric calibration and the online estimation of camera exposure time. We conduct more extensive experiments on public and private datasets to compare our proposed system against other state-of-the-art SLAM systems. Quantitative and qualitative results show that R<sup>3</sup>LIVE++ has significant improvements over others in both accuracy and robustness. Moreover, to demonstrate the extendability of R<sup>3</sup>LIVE++, we developed several applications based on our reconstructed maps, such as high dynamic range (HDR) imaging, virtual environment exploration, and 3D video gaming. Lastly, to share our findings and make contributions to the community, we release our codes, hardware design, and dataset on our Github: [github.com/hku-mars/r3live](https://github.com/hku-mars/r3live).

**Index Terms**—SLAM, 3D reconstruction, State estimation, Sensor fusion

## 1 INTRODUCTION

SIMULTANEOUS localization and mapping (SLAM) is a technology that utilizes a sequence of sensor (e.g., camera, LiDAR, IMU, etc.) data to estimate the sensor poses and simultaneously reconstruct the 3D map of surrounding environments. Since SLAM can estimate poses in real-time, it has been widely applied in localization and feedback control for autonomous robots (e.g., unmanned aerial vehicles [1], [2], automated ground vehicles [3], [4], [5], and self-driving cars [6], [7], [8]). Meanwhile, with the capacity to reconstruct the map in real-time, SLAM is also crucial in various robots navigation, virtual and augmented reality (VR/AR), surveying, and mapping applications. Different applications usually require a different level of mapping details: sparse feature map, 3D dense point cloud map, and 3D radiance map (i.e., a 3D point cloud map with radiance

information). For example, the sparse visual feature map is suitable and has been widely used for camera-based localization, where the sparse features observed in images can be used for calculating the camera's pose [9], [10]. The 3D dense point cloud can capture the geometrical structure of the environment even for tiny objects. Hence it is widely used in robot navigation and obstacle avoidance [2], [11]. Finally, radiance maps containing both geometry and radiance information are used in mobile mapping, AR/VR, video gaming, 3D simulation, and surveying. These applications require both geometric structures and textures to provide virtual environments alike the real world [12], [13].

Existing SLAM systems can be mainly categorized into two classes based on the used sensor: visual SLAM and LiDAR SLAM. Visual SLAM is based on low-cost and SWaP (size, weight, and power)-efficient camera sensors and has achieved satisfactory results in localization accuracy. The rich colorful information measured by cameras also makes the reconstructed map suitable for human interpretation. However, due to the lack of direct accurate depth measurements, the mapping accuracy and resolution of visual

• J. Lin and F. Zhang are with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong SAR, China.  
E-mail: {jiarong.lin, fuzhang}@hku.hk

SLAM are usually lower than LiDAR SLAM. To be more specific, visual SLAM maps the environments by triangulating disparities from multi-view images (e.g., structure from motion for mono-camera, stereo-vision for stereo-camera), an extremely computationally expensive process that often requires hardware acceleration or server clusters. Moreover, limited by the measurement noises and the baseline of multi-view images, the computed depth accuracy drops quadratically with the measurement distance, making visual SLAM difficult to reconstruct large-scale outdoor scenes. Furthermore, visual SLAM can only work in scenarios with good illuminations and will degenerate in high-occlusion or texture-less environments.

On the other hand, LiDAR SLAM is based on LiDAR sensors. Benefiting from the high measurement accuracy (a few centimeters) and the long measurement range (hundreds of meters) of LiDAR sensors, LiDAR SLAM can achieve much higher accuracy and efficiency on both localization and map reconstruction than visual SLAM. However, LiDAR SLAM easily fails in scenarios with insufficient geometry features, such as in long tunnel-like corridors, facing a single big wall, etc. Moreover, LiDAR SLAM can only reconstruct the geometric structure of the environment, but lacks the color information.

Fusing both LiDAR and camera measurements in the SLAM could overcome the degeneration issues of each sensor in localization and produce an accurate, textured, and high-resolution 3D map that suffices the needs of various mapping applications. Motivated by this, we propose R<sup>3</sup>LIVE++, which has the following features:

- It is a LiDAR-Inertial-Visual fusion framework that tightly couples two subsystems: the LiDAR-inertial odometry (LIO) subsystem and the visual-inertial odometry (VIO) subsystem. The two subsystems jointly and incrementally build a 3D radiance map of the environment in real-time. In particular, the LIO subsystem reconstructs the geometric structure by registering new points in each LiDAR scan to the map, and the VIO subsystem recovers the radiance information by rendering pixel colors in each image to points in the map.
- It has a novel VIO design, which tracks the camera pose (and estimates other system state) by minimizing the radiance difference between points from the radiance map and a sparse set of pixels in the current image. The frame-to-map alignment effectively lowers the odometry drift, and the direct photometric error on a sparse set of individual pixels effectively constrains the computation load. Moreover, based on the photometric errors, the VIO is able to estimate the camera exposure time online, which enables the recovery of environments' true radiance information.
- It is extensively validated in real-world experiments in terms of localization accuracy, robustness, and radiance map reconstruction accuracy. Benchmark results on 25 sequences from an open dataset (the NCLT-dataset) show that R<sup>3</sup>LIVE++ achieves the highest overall accuracy among all other state-of-the-art SLAM systems (e.g., LVI-SAM, LIO-SAM, FAST-LIO2, etc) under comparison. The evaluations on

our private dataset show that R<sup>3</sup>LIVE++ is robust to extremely challenging scenarios that LiDAR and/or camera measurements degenerate (e.g., when the device is facing a single texture-less wall). Finally, compared with other counterparts, R<sup>3</sup>LIVE++ estimates the camera exposure time more accurately and reconstructs the true radiance information of the environment with significantly smaller errors when compared to the measured pixels in images.

- It is, to our best knowledge, the first radiance map reconstruction framework that can achieve real-time performance on a PC equipped with a standard CPU without any hardware or GPU accelerations. The system is completely open sourced to ease the reproduction of this work and benefit the following-up researches. Based on a set of offline utilities for mesh reconstruction and texturing further developed, the system shows high potentials in a variety of real-world applications, such as 3D HDR imaging, physics simulation, and video gaming.

## 2 RELATED WORKS

In this chapter, we review existing works related to our method or system, including LiDAR SLAM, visual SLAM, and LiDAR-visual fused SLAM. Due to the large number of existing works, any attempts to give a full review would be incomplete, hence we only select the most relevant ones of each branch for review.

### 2.1 LiDAR(-inertial) SLAM

In recent years, with the rapid development of LiDAR technologies, the reliability and performance of LiDAR sensors have been greatly improved while the cost significantly lowered, drawing increasing amount of researches on LiDAR SLAM [14]. Zhang *et al.* propose a real-time LiDAR odometry and mapping framework, LOAM [15], which achieves localization by scan-to-scan point registration and mapping by scan-to-map registration. In both registrations, only edge and plane feature points are considered to lower the computation load. To make the algorithm run in real-time at computation-limited platforms, Shan *et al.* [16] propose a lightweight and ground-optimized LOAM (LeGO-LOAM), which discards unreliable features in the step of ground plane segmentation. These works [15], [16] are mainly based on multi-line spinning LiDARs. For emerging solid-state LiDARs that have irregular scanning and very small FoV, our previous works [14], [17] use direct scan-to-map registration to achieve localization and mapping.

To further improve the accuracy and robustness of LiDAR SLAM systems, many frameworks that fuse LiDAR measurements with inertial sensors have been proposed. In LOAM [15], an IMU could be used to de-skew the LiDAR scan and give a motion prior for the scan-to-scan registration. It is a loosely-coupled method since the IMU bias (and the full state vector) is not involved in the scan registration process. Compared with loosely-coupled methods, tightly-coupled methods show higher robustness and accuracy, therefore drawing increasing research interests recently. Authors in [18] propose LIOM, which uses a graph

optimization based on priors from LiDAR-Inertial odometry and a rotation-constrained refinement method. Compared with the former algorithms, LIO-SAM [19] optimizes a sliding window of keyframe poses in a factor graph to achieve higher accuracy. Similarly, Li *et al.* propose LiLi-OM [20] for both multi-line and solid-state LiDARs based on a sliding window optimization technique. LINS [21] is the first tightly-coupled LIO that solves the 6 DOF ego-motion via iterated Kalman filtering. To lower the high computation load in calculating the Kalman gain, FAST-LIO [22] proposes a new formula for the Kalman gain computation. The resultant computation complexity depends on the state dimension instead of measurement dimension. Its successor FAST-LIO2 [23] further improves the computation efficiency by proposing an incremental k-d tree. Such a data structure can significantly reduce the time cost of nearest points search and allow the registration of raw points (instead of feature points, such as planes and edges, in the past works). The method using raw points is termed as a “direct” method and could exploit subtle features in the environment and hence increase the localization accuracy and robustness.

The LIO subsystem of R<sup>3</sup>LIVE++ is largely based on FAST-LIO2 [23] since it achieves the best overall performance among its counterparts in terms of accuracy, efficiency, and robustness. Moreover, to address the LiDAR degeneration problem and further improve the localization accuracy, we fuse the LIO subsystem with our VIO subsystem in a tightly-coupled manner.

## 2.2 Visual(-inertial) SLAM

Depending on how a camera measurement is formulated, we review the works of visual SLAM by categorizing them into two branches based on the criteria proposed in [24]: indirect and direct. These two types of methods have very different pipelines: the former one (indirect method) includes feature extraction, data association, and minimization of feature re-projection error. In contrast, the latter one (direct method) directly minimizes the photometric error (or intensity discrepancy) between consecutive images.

Indirect visual SLAM is also called the feature-based method, which has a quite long history. MonoSLAM [25] proposed by Davison *et al.* is the first monocular visual SLAM, which recovers the 3D trajectory of a camera in real-time by creating a sparse but persistent map of natural landmarks within a probabilistic framework. PTAM [26] proposed by Klein and Murray split the tracking and mapping in parallel threads. Visual landmarks in the map are selected from only a few frames to allow efficient bundle-adjustment (BA) optimization that estimates the camera pose and landmark position. Following this idea, a more complete and reliable framework ORB-SLAM [27] was proposed. ORB-SLAM utilizes the same feature (i.e., ORB feature) for all the involved tasks, including tracking, mapping, relocalization, and loop closing. Its further work ORB-SLAM2 [28] improves the accuracy by utilizing the metric scale provided by stereo or RGB-D cameras. The scale issue in pure visual SLAM can also be addressed by fusing inertial sensor data, such as VINS-mono [29] and ORB-SLAM3 [30], which achieve high-accuracy localization by fusing the IMU measurements and image features in a sliding window bundle adjustment optimization.

Direct visual SLAM is also called photometric-based method, which minimizes the intensity differences rather than a geometric error. It has been successfully applied in 2D sparse feature tracking (e.g., Lucas–Kanade optical flow [31]) and then extended to visual SLAM. LSD-SLAM [32] proposed by Engel *et al.* is a direct monocular SLAM algorithm with both tracking and mapping directly operating on image intensities. It incrementally tracks the camera pose using direct image alignment and simultaneously performs a pose graph optimization to keep the entire camera trajectory globally consistent. In DSO [24], authors proposed a fully direct probabilistic model that integrates a full photometric calibration. By incorporating a photometric bundle adjustment, the system outperforms other state-of-the-art works in terms of both accuracy and robustness. To achieve real-time performance on a standard CPU, the authors also exploit the sparsity structures of the corresponding Hessian matrix. While the photometric model provides accurate pose estimation over short-term tracking without data association, the geometric model gives robustness for a large baseline. Hybrid approaches that use both photometric and geometric errors have been proposed, with the most representative work SVO [33] proposed by Forster *et al.*, where the short-term tracking is solved by minimizing the photometric error, while the long-term drift is constrained by a windowed bundle adjustment on visual landmarks.

There have been many discussions in the literature to answer the question: *Which is better?* While it is difficult to answer this question now, it is true that the direct method often shows better short-term performance in low-textured environments [24], [30]. Besides, the direct method is often more computation efficient due to the removal of feature extraction [33]. To leverage these advantages, R<sup>3</sup>LIVE++ uses a photometric-based VIO subsystem. Unlike the pure visual (or visual-inertial) direct SLAM systems, which perform bundle adjustment on photometric errors [24] or feature reprojection errors [33] to restrain long-term drift, the VIO in R<sup>3</sup>LIVE++ makes fully use of the geometry structure reconstructed from LiDAR point cloud by minimizing the radiance errors between map points and image pixels. Such a frame-to-map alignment effectively lowers the odometry drift at a low computation cost. Moreover, pure visual (or visual-inertial) direct methods construct photometric errors on dense images [32] or a sparse set of image patches [24], [33], while the photometric errors of R<sup>3</sup>LIVE++ VIO subsystem are on a sparse set of individual pixels. Furthermore, the VIO in R<sup>3</sup>LIVE++ accounts for the camera photometric calibration (e.g., non-linear response function and lens vignetting) and estimates the camera exposure time online, which help improve the odometry accuracy and recover the true radiance information of the environment.

## 2.3 LiDAR-visual fused SLAM

On the basis of LiDAR-inertial methods, LiDAR-inertial-visual odometry incorporating measurements from visual sensors shows higher robustness and accuracy. Zhang and Singh in [34] propose a LiDAR-inertial-visual system that uses a loosely-coupled VIO as the motion model to initialize the LiDAR mapping subsystem. Similarly, Shao *et al.*

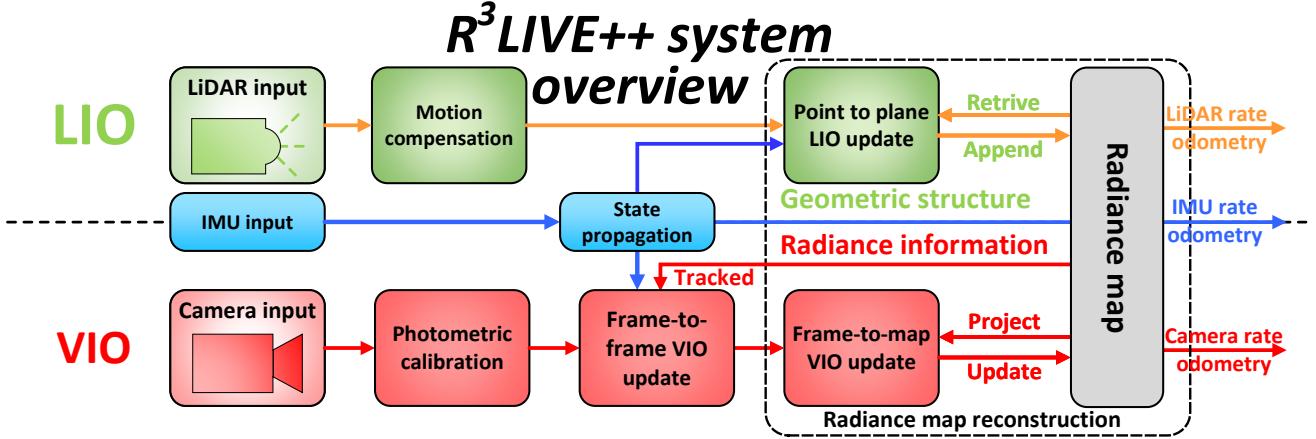


Fig. 2 – The overview of our proposed system.

in [35] propose a stereo visual-inertial LiDAR SLAM that incorporates the tightly-coupled stereo VIO with LiDAR mapping and LiDAR-enhanced visual loop closure. The overall system is still a loosely-coupled fusion since the LiDAR data are not jointly optimized along with the visual-inertial measurements.

There are also some RGB-D-inertial SLAM systems, such as [35], [36]. Designed for RGB-D cameras, these methods are difficult to be applied on LiDARs due to the significant differences in the measurement pattern, range and density between RGB-D cameras and LiDAR sensors. In [37], LiDAR measurements are used to provide depth information for camera images at each frame, forming a system similar to RGB-D camera and hence can leverage existing visual SLAM works such as ORB-SLAM2 [28]. This is also a loosely-coupled method as it ignores the direct constraints imposed by LiDAR measurements.

For works mentioned above, the measurement of LiDAR and camera are fused in a loosely-coupled manner. To achieve higher accuracy and robustness, frameworks that fuse sensor data in a tightly-coupled way are proposed in recent years. Zuo *et al.* [38] propose a LIC-fusion framework combining IMU measurements, sparse visual features, and LiDAR plane and edge features with online spatial and temporal calibration based on the MSCKF framework. The system exhibits higher accuracy and robustness than other state-of-the-art methods in their experiment results. Later on, their further work termed LIC-Fusion 2.0 [39] refines a novel plane-feature tracking algorithm across multiple LiDAR scans within a sliding window to make LiDAR scan matching more robust. Shan *et al.* in [40] propose LVI-SAM that fuses the LiDAR, visual and inertial sensors in a tightly-coupled smooth and mapping framework, which is built atop a factor graph. The LiDAR-inertial and visual-inertial subsystems of LVI-SAM can function independently when failure is detected in one of them, or jointly when enough features are detected. A similar tightly-coupled system is our previous work R<sup>2</sup>LIVE [41], which fuses the LiDAR and camera measurements in an on-manifold iterated Kalman filter. R<sup>2</sup>LIVE can run in various challenging scenarios even with small LiDAR FoV, aggressive motions, sensor failures, and narrow tunnel-like environments with moving objects.

The above LiDAR-inertial-visual systems all use feature-

based methods in both LIO and VIO. In contrast, R<sup>3</sup>LIVE++ uses direct methods in both LIO and VIO to best exploit any subtle features in the environments even in case of extreme scenarios (e.g., structure-less and/or texture-less environments). Moreover, the above LiDAR-inertial-visual systems mainly focus on the localization part and has very limited consideration on the mapping efficiency and accuracy. Hence, their visual and LiDAR subsystem often maintains two separate map for the LIO and VIO, preventing the data fusion at a deeper level and the reconstruction of high-accuracy colored 3D maps. R<sup>3</sup>LIVE++ is designed to perform both localization and radiance map reconstruction in real-time. The central of these two tasks is a single radiance map shared among and maintained by both LIO and VIO. In particular, the LIO subsystem reconstructs the geometric structure of the map and the VIO subsystem recovers the radiance information of the map.

This paper is an extension of the previously published work R<sup>3</sup>LIVE [42]. The extended works of this paper include (1) a full incorporation of the camera photometric calibration, which corrects the camera nonlinear response function and lens vignetting effect; (2) online estimation of the camera exposure time. The estimated exposure time and the camera photometric calibration enables the system to recover the true radiance information of the environment; (3) a more comprehensive evaluation of the system on both open and private dataset in terms of localization accuracy, robustness and radiance map reconstruction accuracy; and (4) release of the system codes, private dataset, and the in-house designed hardware devices for collecting this dataset.

### 3 BASIC MODELS

#### 3.1 Notations

In this paper, we use notations shown in Table 1.

#### 3.2 System overview

To simultaneously estimate the sensor pose and reconstruct the environment radiance map, we design a tightly-coupled LiDAR-inertial-visual sensor fusion framework, as shown in Fig. 2. The proposed framework contains two subsystems: the LIO subsystem (upper part) and the VIO subsystem

TABLE 1 – NOMENCLATURE

Notation	Explanation
	Expressions
$\boxplus/\boxminus$	The encapsulated “boxplus” and “boxminus” operations on manifold [43]
$G(\cdot)$	The value of $(\cdot)$ expressed in global frame
$C(\cdot)$	The value of $(\cdot)$ expressed in camera frame
$\text{Exp}(\cdot)/\text{Log}(\cdot)$	The Rodrigues’ transformation between the rotation matrix and rotation vector
$\delta(\cdot)$	The estimated error of $(\cdot)$ parameterized in tangent space.
$\Sigma(\cdot)$	The covariance matrix of vector $(\cdot)$
	Variables
$b_g, b_a$	The bias of gyroscope and accelerometer in an IMU
${}^Gg$	The gravitational acceleration in global frame
${}^Gv$	The linear velocity in global frame
$({}^G\mathbf{R}_I, {}^G\mathbf{p}_I)$	The IMU attitude and position w.r.t. global frame
$({}^I\mathbf{R}_C, {}^I\mathbf{p}_C)$	The extrinsic between camera and IMU
$\mathbf{x}$	The ground-true state
$\hat{\mathbf{x}}$	The prior estimation of $\mathbf{x}$
$\tilde{\mathbf{x}}$	The current estimate of $\mathbf{x}$ in each ESIKF iteration

(lower part). The LIO subsystem constructs the geometric structure of the radiance map by registering point cloud measurements of each input LiDAR scan. The VIO subsystem recovers the radiance information of the map in two steps: the frame-to-frame VIO update estimates the system state by minimizing the frame-to-frame PnP reprojection error, while the frame-to-map VIO update minimizes the radiance error between map points and the current image. The two subsystems are tightly coupled within an on-manifold error-state iterated Kalman filter framework (ESIKF) [43], where the LiDAR and camera visual measurements are fused to the same system state (Section 3.4) at their respective data reception time (Section 4 and Section 5).

### 3.2.1 Point

Our radiance map is composed of map points in the global frame, each point  $\mathbf{P}$  is a structure as below:

$$\mathbf{P} = [{}^G\mathbf{p}_x, {}^G\mathbf{p}_y, {}^G\mathbf{p}_z, \gamma_r, \gamma_g, \gamma_b]^T = [{}^G\mathbf{p}^T, \boldsymbol{\gamma}^T]^T \in \mathbb{R}^6 \quad (1)$$

where the head sub-vector  ${}^G\mathbf{p} = [{}^G\mathbf{p}_x, {}^G\mathbf{p}_y, {}^G\mathbf{p}_z]^T \in \mathbb{R}^3$  denotes the point 3D position, and the tail sub-vector  $\boldsymbol{\gamma} = [\gamma_r, \gamma_g, \gamma_b]^T \in \mathbb{R}^3$  is the point radiance consisting of three independent channels (i.e., red, green, and blue channel) accounting for the camera photometric calibration (see Section 3.3). Besides, we also record other necessary information of this point, such as the  $3 \times 3$  matrix  $\boldsymbol{\Sigma}_{\mathbf{p}}$  and  $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}}$ , which denote the covariance of the estimation errors of  ${}^G\mathbf{p}$  and  $\boldsymbol{\gamma}$ , respectively, and the timestamps when this point was created and updated.

### 3.2.2 Voxel

To inquiry a point in the radiance map efficiently (e.g., for camera pose tracking in Section 5.3 and map point radiance recovery in Section 5.4), we put map points in fix-size (e.g.  $0.1 \text{ m} \times 0.1 \text{ m} \times 0.1 \text{ m}$ ) voxels. If a voxel has points appended recently (e.g. in recent 1 second), we mark this voxel as *activated*. Otherwise, this voxel is marked as *deactivated*.

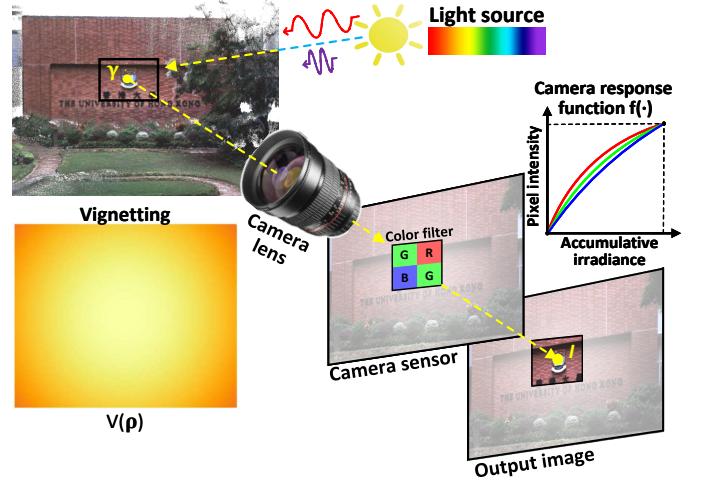


Fig. 3 – The image formation process of a color camera.

### 3.3 Color camera photometric model

A camera observes the radiance of the real world in the form of images that consists of 2D arrays of pixel intensities. In our work, we model the image formation process of a camera based on [44] and further extend the gray camera model to a color camera. As shown in Fig. 3, for a point  $\mathbf{P}$  in the world, it reflects the incoming lights emitted from a light source (e.g., the sun). The reflected lights then pass through the camera lens and finally arrive at the CMOS sensor, which records the intensity of the reflected lights and creates a pixel channel in the output image. The recorded intensity is determined by the *radiance*, a measure of the power reflected at the point  $\mathbf{P}$ .

To model the above imaging process, we denote  $\gamma$  the radiance at point  $\mathbf{P}$ . Since a color camera has three channels in its CMOS sensor: red, green, and blue, the radiance  $\gamma$  has three components:  $\gamma_r, \gamma_g, \gamma_b$ , respectively. For each channel  $i$ , the lights passing through the camera lens has power

$$\mathbf{O}_i(\rho) = V(\rho)\gamma_i \quad (2)$$

where  $V(\rho) \in [0, 1]$  is called the vignetting factor accounting for the lens vignetting effect. Since the vignetting effect is different at different area of the lens, the vignetting factor  $V(\rho)$  is a function of the pixel location  $\rho$ .

$\mathbf{O}_i(\rho)$  is the amount of power that can be received by the sensor and is called the irradiance. When taking an image, the captured irradiance  $\mathbf{O}(\rho)$  is integrated over time (i.e., the exposure time  $\tau$ ). The accumulated irradiance  $\theta_i = \tau V(\rho)\gamma_i$  is then converted as the output of pixel intensity  $\mathbf{I}_i(\rho)$  via the camera response function (CRF)  $f_i(\cdot)$ :

$$\mathbf{I}_i(\rho) = f_i(\tau V(\rho)\gamma_i), \quad \mathbf{I}_i \in [0, 255]. \quad (3)$$

Since a real camera sensor has a limited dynamic range and that the physical scale of the radiance  $\gamma$  can not be recovered anyway, the pixel intensities can be normalized within  $[0, 1]$  without loss of generality.

As noted in (3), different channels often has different non-linear response function (CRF)  $f_i(\cdot)$  and they can be calibrated offline along with the vignetting factor  $V(\rho)$  based on the method in [45]. The exposure time  $\tau$  is estimated online in our work. With the calibration and estimation

results, the radiance of point  $\mathbf{P}$  from the observed pixel value  $\mathbf{I}(\rho)$  can be computed as:

$$\gamma_i = \frac{\mathbf{f}_i^{-1}(\mathbf{I}_i(\rho))}{\tau V(\rho)} \quad (4)$$

*Remark:* Under the assumption of constant continuous light sources and a Lambertian reflection model, the radiance at point  $\mathbf{P}$  is a constant physical value that is invariant to the camera pose. Such invariance to time and camera pose enables us to infer the camera ego-motion from the radiance difference between the map and the current image (with photometric calibration), as detailed in Section 5.3.

### 3.4 State

In our work, we define the full state  $\mathbf{x}$  as:

$$\mathbf{x} = ({}^G\mathbf{R}_I, {}^G\mathbf{p}_I, {}^G\mathbf{v}, \mathbf{b}_g, \mathbf{b}_a, {}^G\mathbf{g}, {}^I\mathbf{R}_C, {}^I\mathbf{p}_C, \epsilon, {}^I\mathbf{t}_C, \phi) \quad (5)$$

where the notations  ${}^G\mathbf{R}_I$ ,  ${}^G\mathbf{p}_I$ ,  ${}^G\mathbf{v}$ ,  $\mathbf{b}_g$ ,  $\mathbf{b}_a$ ,  ${}^G\mathbf{g}$ ,  ${}^I\mathbf{R}_C$ ,  ${}^I\mathbf{p}_C$  are explained in Table 1,  ${}^I\mathbf{t}_C$  is the time-offset between IMU and camera while LiDAR is assumed to be synced with the IMU already,  $\epsilon = 1/\tau$  is the inverse camera exposure time,  $\phi = [f_x, f_y, c_x, c_y]^T$  are the camera intrinsics, where  $(f_x, f_y)$  denote the camera focal length and  $(c_x, c_y)$  the offsets of the principal point from the top-left corner of the image plane. The camera extrinsic  $({}^I\mathbf{R}_C, {}^I\mathbf{p}_C)$ , intrinsic  $\phi$  and time-offset  ${}^I\mathbf{t}_C$  would usually have their rough values available (e.g., offline calibration, CAD model, manufacturer's manual). To cope with the possible calibration errors (e.g., extrinsic  $({}^I\mathbf{R}_C, {}^I\mathbf{p}_C)$  and intrinsic  $\phi$ ) or online drifting (e.g., time-offset  ${}^I\mathbf{t}_C$ ), we also include them in the state  $\mathbf{x}$  such that they will be estimated online. Beside, we also estimate the camera exposure time online in order to recover the true radiance value of each map point.

## 4 LIDAR-INERTIAL ODOMETRY (LIO)

Our LIO subsystem reconstructs the geometry structure of the environment by registering each new LiDAR scan to the global map. We use the generalized-iterative closet point (GICP) method [46] to iteratively estimate the LiDAR pose (and other system state) by minimizes the distance of each point in the scan to a plane fitted from the corresponding points in the map. The estimated state estimate is then used to append the new points to the map.

### 4.1 LiDAR point-to-plane residual

As shown in Fig. 2, our LIO subsystem constructs the geometric structure of the global map. For the  $k$ -th input LiDAR scan, we first compensate the in-frame motion with a IMU backward propagation introduced in [22]. Let  $\mathcal{L}_k = \{{}^L\mathbf{p}_1, \dots, {}^L\mathbf{p}_m\}$  be the set of  $m$  LiDAR points after motion compensation, we compute the residual of each raw point (or a downsampled subset) of  ${}^L\mathbf{p}_s \in \mathcal{L}_k$  where  $s$  is the index of point and the superscript  $L$  denotes that the point is represented in the LiDAR-reference frame.

With  $\check{\mathbf{x}}_k$  being the estimate of  $\mathbf{x}_k$  at the current iteration, we transform  ${}^L\mathbf{p}_s$  from LiDAR frame to the global frame:

$${}^G\mathbf{p}_s = {}^G\check{\mathbf{R}}_{I_k}({}^I\mathbf{R}_L {}^L\mathbf{p}_s + {}^I\mathbf{p}_L) + {}^G\check{\mathbf{p}}_{I_k} \quad (6)$$

To register the point to the global map, we search for the nearest five points in the map. To accelerate the nearest neighbor search, map points are organized into an incremental k-d tree (see [23]). The found nearest neighbor points are used to fit a plane with normal  $\mathbf{u}_s$  and centroid  $\mathbf{q}_s$ , then the LiDAR measurement residual  $\mathbf{r}_l(\check{\mathbf{x}}_k, {}^L\mathbf{p}_s)$  is:

$$\mathbf{r}_l(\check{\mathbf{x}}_k, {}^L\mathbf{p}_s) = \mathbf{u}_s^T ({}^G\mathbf{p}_s - \mathbf{q}_s) \quad (7)$$

### 4.2 LIO ESIKF update

The residual in (7) should be zero ideally. However, due to the estimation error in  $\check{\mathbf{x}}_k$  and the LiDAR measurement noise, this residual is often not zero and can be used to refine the state estimate  $\check{\mathbf{x}}_k$ . Specifically, let  $\mathbf{n}_s$  be the measurement noise of the point  ${}^L\mathbf{p}_s$ , we have the relation between the true point location  ${}^L\mathbf{p}_s^{\text{gt}}$  and the measured one  ${}^L\mathbf{p}_s$  as below:

$${}^L\mathbf{p}_s = {}^L\mathbf{p}_s^{\text{gt}} + \mathbf{n}_s, \mathbf{n}_s \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{n}_s}). \quad (8)$$

This true point location together with the true state  $\mathbf{x}_k$  should lead to zero residual in (7), i.e.,

$$\mathbf{0} = \mathbf{r}_l(\mathbf{x}_k, {}^L\mathbf{p}_s^{\text{gt}}) \approx \mathbf{r}_l(\check{\mathbf{x}}_k, {}^L\mathbf{p}_s) + \mathbf{H}_s^l \delta \check{\mathbf{x}}_k + \boldsymbol{\alpha}_s, \quad (9)$$

where  $\mathbf{x}_k$  is parameterized by its error  $\delta \check{\mathbf{x}}_k$  in the tangent space of  $\check{\mathbf{x}}_k$  (i.e.,  $\mathbf{x}_k = \check{\mathbf{x}}_k \boxplus \delta \check{\mathbf{x}}_k$ ),  $\boldsymbol{\alpha}_s \sim \mathcal{N}(\mathbf{0}, \Sigma_{\boldsymbol{\alpha}_s})$  is the lumped noise due to  $\mathbf{n}_s$ , and  $\mathbf{H}_s^l$  is the Jacobian of the residual w.r.t.  $\delta \check{\mathbf{x}}_k$ .

Equation (9) constitutes an observation distribution for  $\mathbf{x}_k$  (or equivalently  $\delta \check{\mathbf{x}}_k \triangleq \mathbf{x}_k \boxminus \check{\mathbf{x}}_k$ ), which is combined with the prior distribution from the IMU propagation:

$$\min_{\delta \check{\mathbf{x}}_k} \left( \|(\check{\mathbf{x}}_k \boxplus \delta \check{\mathbf{x}}_k) \boxminus \hat{\mathbf{x}}_k\|_{\Sigma_{\delta \check{\mathbf{x}}_k}}^2 + \sum_{s=1}^m \|\mathbf{r}_l(\check{\mathbf{x}}_k, {}^L\mathbf{p}_s) + \mathbf{H}_s^l \delta \check{\mathbf{x}}_k\|_{\Sigma_{\boldsymbol{\alpha}_s}}^2 \right) \quad (10)$$

where  $\|\mathbf{x}\|_{\Sigma}^2 = \mathbf{x}^T \Sigma^{-1} \mathbf{x}$  is the squared Mahalanobis distance with covariance  $\Sigma$ ,  $\hat{\mathbf{x}}_k$  is the IMU propagated state estimate, and  $\Sigma_{\delta \check{\mathbf{x}}_k}$  is the IMU propagated state covariance. The detailed derivation of first item in (10) can be found in Section IV-E of R<sup>2</sup>LIVE [41].

Solving (10) leads to the Maximum A-Posteriori (MAP) estimate of  $\delta \check{\mathbf{x}}_k^o$  which is then added to  $\check{\mathbf{x}}_k$  as below

$$\check{\mathbf{x}}_k \leftarrow \check{\mathbf{x}}_k \boxplus \delta \check{\mathbf{x}}_k^o \quad (11)$$

The above iteration process is iterated until convergence (i.e., the update  $\delta \check{\mathbf{x}}_k^o$  is smaller than a given threshold). The converged state estimate  $\check{\mathbf{x}}_k$  is then used as the starting point of the IMU propagation until the reception of the next LiDAR scan or camera image. Furthermore, the converged estimate  $\check{\mathbf{x}}_k$  is used to append points in the current LiDAR scan to the global map as follows. For the  $s$ -th point  ${}^L\mathbf{p}_s \in \mathcal{L}_k$ , its position in global frame  ${}^G\mathbf{p}_s$  is first obtained by (6). If  ${}^G\mathbf{p}_s$  has nearby points in the map with distance 1 cm (see Section 3.2.1),  ${}^G\mathbf{p}_s$  will be discarded to maintain a spatial resolution of 1 cm. Otherwise, a new point structure  $\mathbf{P}_s$  will be created in the map with:

$$\mathbf{P}_s = [{}^G\mathbf{p}_s^T, \boldsymbol{\gamma}_s^T]^T = [{}^G\mathbf{p}_s, \mathbf{0}]^T \quad (12)$$

where the radiance vector  $\boldsymbol{\gamma}_s$  is set as zero and will be initialized at the first time it is observed in forthcoming images (see Section 5.4). Finally, we mark the voxel containing  ${}^G\mathbf{p}_s$  as *activated* such that the radiance of points in this voxel can be updated by the forthcoming images (see Section 5.4).

## 5 VISUAL-INERTIAL ODOMETRY (VIO)

While our LIO subsystem reconstructs the geometric structure of the environment, our VIO subsystem recovers the radiance information from the input color images. To be more specific, our VIO subsystem projects a certain number of points (i.e., tracked points) from the global map to the current image, then it iteratively estimates the camera pose (and other system state) by minimizing the radiance error of these points. Only a sparse set of tracked map points is used for the sake of computation efficiency.

Our proposed framework is different from previous photometric-based methods [33], [47], which constitute the residual of a point by considering the photometric error over all its neighborhood pixels (i.e., a patch). These patch-based methods achieve stronger robustness and faster convergence speed than that without. However, the patch-based method is not invariant to either translation or rotation, which requires estimating the relative transform when aligning one patch to another. Plus, the calculation of the residual is not completely precise by assuming the depths of all pixels in the patch are the same as the mid-point. On the other hand, our VIO is operated at an individual pixel, which utilizes the radiance of a single map point to compute the residual. The radiance, which is updated simultaneously in the VIO, is an inherent property of a point in the world and is invariant to both camera translation and rotation. To ensure a robust and fast convergence, we design a two-step pipeline shown in Fig. 2, where in the first step (i.e., frame-to-frame VIO) we leverage a frame-to-frame optical flow to track map points observed in the last frame and obtain a rough estimate of the system’s state by minimizing the Perspective-n-Point (PnP) reprojection error of the tracked points (Section 5.2). Then, in the second step (i.e., frame-to-map VIO), the state estimate is further refined by minimizing the difference between the radiance of map points and the pixel intensities at their projected location in the current image (Section 5.3). With the converged state estimate and the raw input image, we finally update map points radiance according to the current image measurement (Section 5.4).

### 5.1 Photometric correction

For each incoming image  $\mathbf{I}$ , we first correct the image non-linear CRF  $f_i(\cdot)$  and the vignetting factor  $V(\cdot)$ , which are calibrated in advance (see Section 3.3), to obtain the photometrically corrected image  $\Gamma$ , whose  $i$ -th channel at pixel location  $\rho$  is:

$$\Gamma_i(\rho) = \frac{f_i^{-1}(\mathbf{I}_i(\rho))}{V(\rho)}. \quad (13)$$

The photometrically corrected image  $\Gamma$  is then used in the following VIO pipelines including the frame-to-frame VIO, frame-to-map VIO and radiance recovery.

### 5.2 Frame-to-frame Visual-Inertial odometry

#### 5.2.1 Perspective-n-Point reprojection error

Assume we have tracked  $m$  map points  $\mathcal{P} = \{\mathbf{P}_1, \dots, \mathbf{P}_m\}$  in the last image frame  $\mathbf{I}_{k-1}$  with their projected location in  $\mathbf{I}_{k-1}$  being  $\{\rho_{1_{k-1}}, \dots, \rho_{m_{k-1}}\}$ , we leverage the

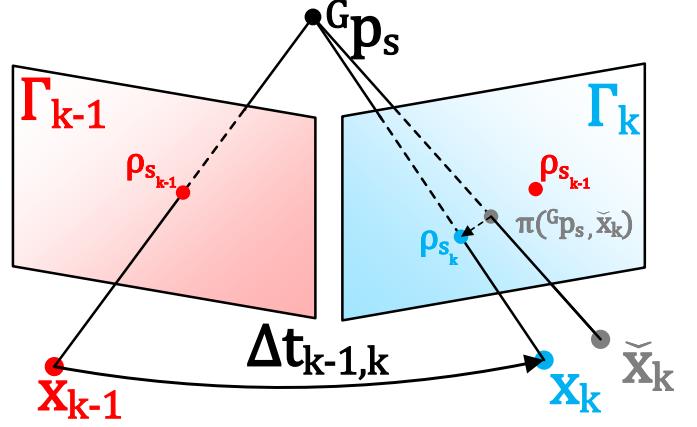


Fig. 4 – Frame-to-frame VIO estimates the system’s state by minimizing the PnP reprojection error of map points observed in the last frame.

Lucas–Kanade optical flow to find out their corresponding location in the current image  $\mathbf{I}_k$ , denoted as  $\{\rho_{1_k}, \dots, \rho_{m_k}\}$ . Then, we iteratively minimize the reprojection errors of  $\mathcal{P}$  to obtain a rough estimate of the state (see Fig. 4). Specifically, taking the  $s$ -th point  $\mathbf{P}_s = [{}^G \mathbf{p}_s^T, \gamma_s^T]^T \in \mathcal{P}$  as example, let  $\check{\mathbf{x}}_k$  be the state estimate at the current iteration, the projection error  $\mathbf{r}_c$  ( $\check{\mathbf{x}}_k, \rho_{s_k}, {}^G \mathbf{p}_s$ ) is

$$\mathbf{r}_c (\check{\mathbf{x}}_k, \rho_{s_k}, {}^G \mathbf{p}_s) = \rho_{s_k} - \pi({}^G \mathbf{p}_s, \check{\mathbf{x}}_k) \quad (14)$$

where  $\pi({}^G \mathbf{p}_s, \check{\mathbf{x}}_k) \in \mathbb{R}^2$  is the predicted pixel location computed as below:

$$\pi({}^G \mathbf{p}_s, \check{\mathbf{x}}_k) = \pi_{ph}({}^G \mathbf{p}_s, \check{\mathbf{x}}_k) + \frac{I \check{\mathbf{t}}_{C_k}}{\Delta t_{k-1,k}} (\rho_{s_k} - \rho_{s_{k-1}}) \quad (15)$$

where the first term  $\pi_{ph}({}^G \mathbf{p}_s, \check{\mathbf{x}}_k)$  is the standard camera pin-hole model, the second one is the temporal correction factor [48], and  $\Delta t_{k-1,k}$  is the time interval between the last and current image.

#### 5.2.2 Frame-to-frame VIO Update

Similar to the LIO update, the state estimation error in  $\check{\mathbf{x}}_k$  and the camera measurement noise will lead to a certain residual in (14), from which we can update the state estimate  $\check{\mathbf{x}}_k$  as follows. First, the measurement noise in the residual (14) consists of two sources: one is the pixel tracking error in  $\rho_{s_k}$  and the other lies in the map point location error  ${}^G \mathbf{p}_s$ ,

$${}^G \mathbf{p}_s = {}^G \mathbf{p}_s^{gt} + \mathbf{n}_{\mathbf{p}_s}, \mathbf{n}_{\mathbf{p}_s} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{n}_{\mathbf{p}_s}}) \quad (16)$$

$$\rho_{s_k} = \rho_{s_k}^{gt} + \mathbf{n}_{\rho_{s_k}}, \mathbf{n}_{\rho_{s_k}} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{n}_{\rho_{s_k}}}) \quad (17)$$

where  ${}^G \mathbf{p}_s^{gt}$  and  $\rho_{s_k}^{gt}$  are the true values of  ${}^G \mathbf{p}_s$  and  $\rho_{s_k}$ , respectively. Then, correcting such noises and using the true system state should lead to zero residual, i.e.,

$$\mathbf{0} = \mathbf{r}_c (\mathbf{x}_k, \rho_{s_k}^{gt}, {}^G \mathbf{p}_s^{gt}) \approx \mathbf{r}_c (\check{\mathbf{x}}_k, \rho_{s_k}, {}^G \mathbf{p}_s) + \mathbf{H}_s^r \delta \check{\mathbf{x}}_k + \beta_s \quad (18)$$

where  $\mathbf{H}_s^r$  is the Jacobian of the residual w.r.t.  $\delta \check{\mathbf{x}}_k$  and  $\beta_s \sim \mathcal{N}(\mathbf{0}, \Sigma_{\beta_s})$  is the lumped noise due to  $\mathbf{n}_{\mathbf{p}_s}$  and  $\mathbf{n}_{\rho_{s_k}}$ .

Equation (18) constitutes an observation distribution for  $x_k$ , which is combined with the IMU propagation to obtain the MAP estimate of the state in the same way as the LIO

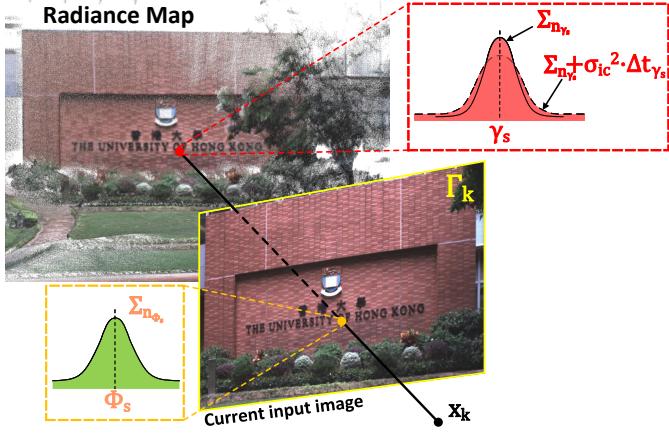


Fig. 5 – Frame-to-map VIO refines the state estimate by minimizing the radiance error between the map point and the observed radiance in the current image.

update detailed in Section 4.2. The converged state estimate is then refined in the frame-to-map VIO in the next section.

*Remark:* Since the camera pin-hole model  $\pi_{ph}(^G p_s, \mathbf{x}_k)$  in (15) is related to camera pose (consisting of the IMU pose ( ${}^G \mathbf{R}_I, {}^G \mathbf{p}_I$ ) and camera extrinsic ( ${}^I \mathbf{R}_C, {}^I \mathbf{p}_C$ )) and intrinsic  $\phi$ , so the projection model  $\pi({}^G \mathbf{p}_s, \mathbf{x}_k)$  is also related to these state components. In addition,  $\pi({}^G \mathbf{p}_s, \mathbf{x}_k)$  is also related to the temporal offset  ${}^I t_C$  due to the temporal correction factor. This will cause  $\mathbf{H}_s^r$  to contain nonzero elements corresponding to the IMU pose ( ${}^G \mathbf{R}_I, {}^G \mathbf{p}_I$ ), camera extrinsic ( ${}^I \mathbf{R}_C, {}^I \mathbf{p}_C$ ), intrinsic  $\phi$ , and temporal offset  ${}^I t_C$ , and hence an update of them in the state estimation.

### 5.3 Frame-to-map Visual-Inertial odometry

#### 5.3.1 Frame-to-map radiance error

The frame-to-frame VIO update can provide a good state estimate  $\check{\mathbf{x}}_k$ , which is further refined by the frame-to-map VIO update by minimizing the radiance error of the tracked map points  $\mathcal{P}$ . Let  $\Gamma_k$  the photometrically calibrated image at the  $k$ -th step (see (13)). With the state estimate at the current iteration,  $\check{\mathbf{x}}_k$ , which contains the estimated camera pose, extrinsic, intrinsic, and exposure time, we project a tracked map point  $\mathbf{P}_s \in \mathcal{P}$  to the image plane to obtain its pixel location  $\check{\rho}_{s_k} = \pi({}^G \mathbf{p}_s, \check{\mathbf{x}}_k)$  (see (15) and Fig. 5). Then, the observed radiance denoted by  $\Phi_s$  can be computed from the exposure time component  $\check{\epsilon}_k$  in  $\check{\mathbf{x}}_k$  as:  $\Phi_s = \check{\epsilon}_k \Gamma_k(\check{\rho}_{s_k})$ . Finally, the frame-to-map radiance error is the difference between the radiance component  $\gamma_s$  of the point  $\mathbf{P}_s$  and the observed value  $\Phi_s$ :

$$\mathbf{r}_c(\check{\mathbf{x}}_k, {}^G \mathbf{p}_s, \gamma_s) = \Phi_s - \gamma_s, \quad \Phi_s = \check{\epsilon}_k \Gamma_k(\check{\rho}_{s_k}), \quad (19)$$

where  $\Phi_s, \Gamma_k(\check{\rho}_{s_k})$  and  $\gamma_s$  both contain three channels: red, green, and blue.

#### 5.3.2 Frame-to-map VIO update

The measurement noise in (19) come from both the component  $\gamma_s$  and  $\Phi_s$ . For the component  $\gamma_s$ , we model it as:

$$\begin{aligned} \gamma_s &= \gamma_s^{gt} + \mathbf{n}_{\gamma_s} + \mathbf{n}_{ic}, \quad \mathbf{n}_{\gamma_s} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{n}_{\gamma_s}}) \\ \mathbf{n}_{ic} &\sim \mathcal{N}(\mathbf{0}, \sigma_{ic}^2 \cdot \Delta t_{\gamma_s}) \end{aligned} \quad (20)$$

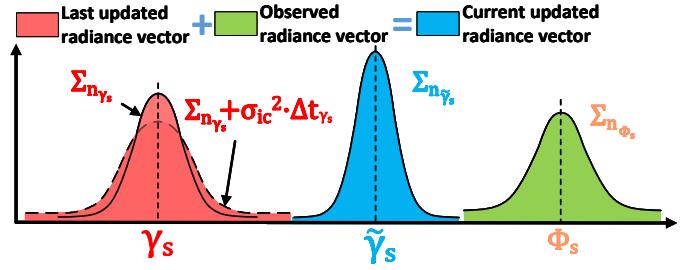


Fig. 6 – We update the radiance  $\gamma_s$  of a map point via Bayesian update.

where  $\gamma_s^{gt}$  is the ground truth of  $\gamma_s$ , the first noise  $\mathbf{n}_{\gamma_s}$  is due to the radiance estimation error (Section 5.4), and the second noise  $\mathbf{n}_{ic}$  accounts for the radiance temporal change caused by illumination change. Since the illumination often changes slowly over time, we model it as a random walk, hence its covariance is linear to  $\Delta t_{\gamma_s}$ , the time interval between current time and last update time of  $\mathbf{P}_s$ .

For the second component  $\Phi_s$  in (19), it is computed from the state estimate  $\check{\mathbf{x}}_k$  and the current image  $\Gamma_k$  as  $\Phi_s = \check{\epsilon}_k \Gamma_k(\pi({}^G \mathbf{p}_s, \check{\mathbf{x}}_k))$ , hence its noise consists of two sources: one is the state estimation error (from  $\check{\mathbf{x}}_k$ ) and the other is the image measurement noise (from  $\Gamma_k$ ):

$$\Phi_s = \Phi_s^{gt} + \mathbf{n}_{\Phi_s}, \quad \mathbf{n}_{\Phi_s} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{n}_{\Phi_s}}) \quad (21)$$

where  $\Sigma_{\mathbf{n}_{\Phi_s}}$  denotes the covariance due to these two noise sources.

Combining (19), (20) and (21), we obtain the first order Taylor expansion of the true zero residual  $\mathbf{r}_c(\mathbf{x}_k, {}^G \mathbf{p}_s^{gt}, \gamma_s^{gt})$ :

$$\mathbf{0} = \mathbf{r}_c(\mathbf{x}_k, {}^G \mathbf{p}_s^{gt}, \gamma_s^{gt}) \approx \mathbf{r}_c(\check{\mathbf{x}}_k, {}^G \mathbf{p}_s, \mathbf{c}_s) + \mathbf{H}_s^c \delta \check{\mathbf{x}}_k + \zeta_s \quad (22)$$

where  $\mathbf{H}_s^c$  is the Jacobian of the residual w.r.t.  $\delta \check{\mathbf{x}}_k$  and  $\zeta_s \sim \mathcal{N}(\mathbf{0}, \Sigma_{\zeta_s})$  is the lumped noise due to noises in  $\gamma_s$  and  $\Phi_s$ .

Similar as before, (22) constitutes an observation distribution for state  $\mathbf{x}_k$ , which is combined with the IMU propagation to obtain the MAP estimate of the state.

*Remark:* Since the  $\Phi_s$  in (19) is related to camera exposure time  $\epsilon$ , it will cause  $\mathbf{H}_s^c$  to contain nonzero elements corresponding to the exposure time and hence an update of them in the state estimation.

### 5.4 Recovery of radiance information

After the frame-to-map VIO update, we have the precise pose of the current image. Then, we perform the Bayesian update to determine the optimal radiance of all map points such that the average radiance error between each point and its viewed images is minimal.

First of all, we retrieve all the points in all activated voxels (activated in Section 4.2). Assume the retrieved point set is  $\mathcal{Q} = \{\mathbf{P}_1, \dots, \mathbf{P}_n\}$ . For the  $s$ -th point  $\mathbf{P}_s = [{}^G \mathbf{p}_s^T, \gamma_s^T]^T \in \mathcal{Q}$  falling in the current image FoV, we first can obtain the observed radiance vector  $\Phi_s$  by (19) and its covariance  $\Sigma_{\mathbf{n}_{\Phi_s}}$  by (21). Then, if  $\mathbf{P}_s$  is a new point appended by the LIO subsystem (see Section 4.2) with  $\gamma_s = \mathbf{0}$ , we set:

$$\gamma_s = \Phi_s, \quad \Sigma_{\mathbf{n}_{\gamma_s}} = \Sigma_{\mathbf{n}_{\Phi_s}} \quad (23)$$

Otherwise, the radiance vector  $\gamma_s$  saved in the map (see (20)) is fused with newly observed radiance vector  $\Phi_s$  with covariance  $\Sigma_{\Phi_s}$  via Bayesian update (see Fig. 6):

$$\Sigma_{n_{\tilde{\gamma}_s}} = \left( (\Sigma_{n_{\gamma_s}} + \sigma_{ic}^2 \cdot \Delta t_{\gamma_s})^{-1} + \Sigma_{n_{\Phi_s}}^{-1} \right)^{-1} \quad (24)$$

$$\tilde{\gamma}_s = \left( (\Sigma_{n_{\gamma_s}} + \sigma_{ic}^2 \cdot \Delta t_{\gamma_s})^{-1} \gamma_s + \Sigma_{n_{\Phi_s}}^{-1} \Phi_s \right)^{-1} \Sigma_{n_{\tilde{\gamma}_s}} \quad (25)$$

$$\gamma_s = \tilde{\gamma}_s, \quad \Sigma_{n_{\gamma_s}} = \Sigma_{n_{\tilde{\gamma}_s}} \quad (26)$$

### 5.5 Update of the tracking points

After the recovery of radiance information, we update the tracked point set  $\mathcal{P}$  for the next frame of image use. Firstly, we remove points from current  $\mathcal{P}$  if their projection error in (14) or radiance error in (19) are too large, and also remove the points which does not fall into the current image FoV. Secondly, we project each point in  $\mathcal{Q}$  to the current image and add it to  $\mathcal{P}$  if no other tracked points already existed in a neighborhood of 50 pixels.

## 6 EXPERIMENTS

In this chapter, we conduct extensive experiments to validate the advantages of our proposed system against other counterparts in threefold: 1) To verify the accuracy in localization, we quantitatively compare our system against existing state-of-the-art SLAM systems on a public dataset (NCLT-dataset). 2) To validate the robustness of our framework, we test it under various challenging scenarios where camera and LiDAR sensor degeneration occurs. 3) To evaluate the accuracy of our system in reconstructing the radiance map, we compare it against existing baselines in estimating the camera exposure time and calculating the average photometric error w.r.t. each image. In the experiments, two datasets are used for evaluation: the NCLT-dataset and the R<sup>3</sup>LIVE-dataset.

### 6.1 NCLT-dataset

To compare the accuracy of our proposed method against other state-of-the-art SLAM systems, we perform quantitative evaluations on NCLT dataset [49]. NCLT-dataset is a large-scale, long-term autonomy dataset for robotics research that was collected on the University of Michigan’s North Campus. The dataset is comprised of 27 sequences that are collected by exploring the campus, both indoors and outdoors, on varying paths, and at different time of a day across all four seasons. Each sequence includes data from the omnidirectional camera, 3D lidar, planar lidar, GPS, and wheel encoders on a Segway robot.

We choose NCLT-dataset for three reasons: 1) NCLT-dataset is currently the largest public dataset with ground-truth trajectories of high quality. 2) NCLT-dataset provides all raw data sampled by the sensors, which meets our requirement for the input data. 3) NCLT-dataset has many challenging scenarios, such as moving obstacles (e.g., pedestrians, bicyclists, and cars), illumination changing, varying viewpoint, seasonal and weather changes (e.g., falling leaves and snow), and long-term structural changes caused by construction projects.

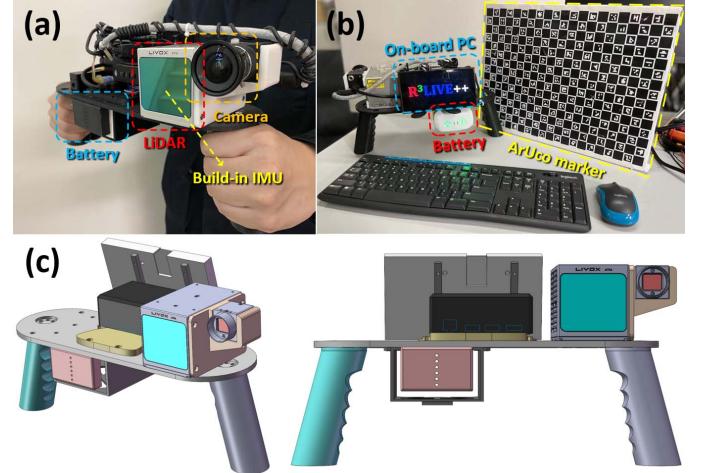
In the experiments, the front-facing camera data (one of five) and the 3D LiDAR data are used for all systems under

evaluation. Moreover, we notice some time synchronization errors in two sequences (i.e., 2012-03-17 and 2012-08-04), where the LiDAR timestamp is 100ms delayed from the IMU timestamp (about one LiDAR-frame). Therefore, we exclude these two sequences from the evaluation. As a result, 25 sequences are evaluated with total traveling length up to 138 km and duration up to 33 h:34 min.

### 6.2 Our private dataset: R<sup>3</sup>LIVE-dataset

While the large-scale NCLT-dataset is suitable for evaluating the localization accuracy, it didn’t cover any scenarios with sensor degeneration, preventing us from evaluating the system robustness, which is one of the major motivations of this work. Moreover, the camera photometric calibration and the ground-true exposure time are not available in the NCLT-dataset, which are essential for the reconstruction of the radiance maps and the evaluation of the online exposure time estimation. To fill this gap, we designed a handheld data collection device and made a new dataset named R<sup>3</sup>LIVE-dataset. The dataset and hardware device are released along with the codes of this work to facilitate the reproduction of our work.

#### 6.2.1 Handheld device for data collection

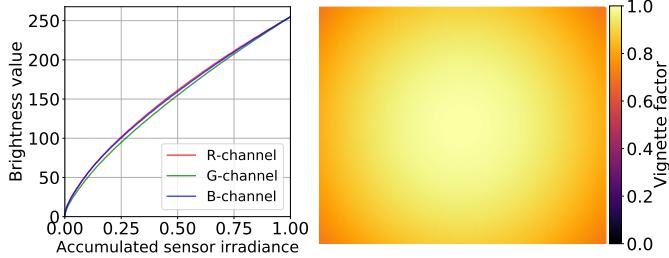


**Fig. 7 –** (a) shows our handheld device for data collection. (b) shows the ArUco marker board to provide the ground-truth for evaluating the system accuracy. (c) shows our open source schematics model.

Our handheld device for data collection is shown in Fig. 7(a), which includes a power supply unit, an onboard computer *DJI manifold-2c* (equipped with an *Intel i7-8550u* CPU and 8 GB RAM), a *FLIR Blackfly BFS-u3-13y3c* global shutter camera, and a *LiVOX AVIA* 3D LiDAR. The camera FoV is  $82.9^\circ \times 66.5^\circ$  and the LiDAR FoV is  $70.4^\circ \times 77.2^\circ$ . To quantitatively evaluate the accuracy of our algorithm (Section 6.5) even in GPS denied environments, we use an ArUco marker [50] as a reference to calculate the sensor pose when returns to the starting point, which enables to evaluate the localization drift. All of the mechanical modules of this device are designed as FDM printable, schematics of the design are opened on our Github: [github.com/hku-mars/r3live](https://github.com/hku-mars/r3live).

To correct the camera’s nonlinear response function and the vignette effect, we perform photometric calibration on

the camera based on the method in [45]. The calibrated results are shown in Fig.8, which are also released on Github.



**Fig. 8** – The left figure shows the calibrated nonlinear response function in three channels (red, green, and blue). The right one plots the calibrated vignetting factors at each image pixel.

### 6.2.2 The $R^3LIVE$ -dataset

The  $R^3LIVE$ -dataset was collected within the campuses of the University of Hong Kong (HKU) and the Hong Kong University of Science and Technology (HKUST). As summarized in Table 2, the dataset includes 13 sequences that are collected by exploring both indoor and outdoor environments, in various scenes (e.g., walkway, park, forest, etc) at different time in a day (i.e., morning, noon, and evening). This allows the dataset to capture both structured urban buildings and cluttered field environments with different lighting conditions. The dataset also includes three sequences (degenerate\_seq\_00/01/02) where the LiDAR or camera (or both) degenerate by occasionally facing the device to a single and/or texture-less plane (e.g., wall, the ground) or visually. The total traveling length reaches 8.4 km, duration reaching 2.4 h. More details of each sequence will be provided in sequel when it is used.

### 6.3 System configurations

For the sake of fair comparison, in the evaluation of our systems and their counterparts, each system uses the same parameters for all sequences in the same dataset. For the counterpart systems (e.g., LIO-SAM, LVI-SAM, FAST-LIO2, etc), we use their default configurations on their Github repository except for some necessary adjustments to match the hardware setup. For our system, we also make its configuration available on Github, “r3live\_config\_nclt.yaml” for NCLT-dataset and “r3live\_config.yaml” for  $R^3LIVE$ -dataset.

### 6.4 Experiment-1: Evaluation of localization accuracy

In this experiment, we benchmark the localization accuracy of our systems against other state-of-the-art SLAM systems, including LIO-SAM [19], LVI-SAM [40], FAST-LIO2 [23], and our previous work  $R^2LIVE$  [41], on the NCLT-dataset [49]. LIO-SAM and FAST-LIO2 are LiDAR-inertial systems without fusing image data (Section 2.1), while LVI-SAM and  $R^2LIVE$  are two feature-based LiDAR-inertial-visual systems (see Section 2.3). Since our work is a state estimator without any loop detection and correction, we

deactivated the loop closure of LIO-SAM and LVI-SAM for the sake of fair comparison. Since the camera photometric calibration of the NCLT-dataset is not available, we disable the photometric calibration modules of our VIO-subsystem by using  $V(\cdot) = 1$  and  $\mathbf{f}_i(\cdot) = 1$ .

Table 3 shows the absolute position error (APE) [51] of these methods, where *Our-LIO* is the LIO subsystem of our system. LIO-SAM and LVI-SAM failed in some sequences, and these sequences are excluded from the computation of the average APE. As can be seen from this table, with the average APE only 8.51 m, our proposed system achieves the best overall performance than feature-based LiDAR-inertial-visual systems  $R^2LIVE$  and LVI-SAM. The performance improvement mainly come from the direct method used in the LIO subsystem and the tight-coupling of the LIO and VIO subsystems, the former can be seen by comparing the direct method *FAST-LIO2* to the feature-based method *LIO-SAM* in Table 3 (and also detailed in [23]), the latter improves the accuracy of the VIO subsystem (hence the complete system) by leveraging the high-accuracy geometry structure reconstructed from the LiDAR. Furthermore, the overall APE of our system is lower than its LIO subsystem *Our-LIO* and the other LIO systems (i.e., *FAST-LIO2* and *LIO-SAM*), which confirms the effectiveness of fusing camera data. Indeed, we found that in the evaluated sequences, the LiDAR sensor may occasionally face the sun, which creates a large number of noisy LiDAR points and adversely affect the LIO accuracy. There are certain sequences where our system is slightly outperformed by its LIO subsystem and the other LIO systems, the reason residues in moving objects (e.g., pedestrians, bicyclists, and cars) which may adversely affect the VIO (hence the overall system). In Fig. 10, we overlay all the 25 ground-truth trajectories (in the left figure) and ours (the right one). As can be seen, the overlaid trajectories estimated by our system agree with the ground-truth well and each trajectory can still be clearly distinguished without noticeable errors. Note that these 25 trajectories are collected on the same campus area across different time in a day and seasons in a year, still our system can produce consistent and reliable trajectory estimation with these illumination and scene changes, demonstrating the robustness of our system.

### 6.5 Experiment-2: Evaluation of robustness

Besides illumination and scene change, we also test the robustness of our system to extreme scenarios where sensor degeneration occurs. We use the  $R^3LIVE$ -dataset, which contains such extreme scenarios.

#### 6.5.1 Evaluation of robustness in LiDAR degenerated scenarios

In this experiment, we evaluate the robustness of our proposed system by testing our system on the sequence “degenerate\_seq\_00” and “degenerate\_seq\_01” of  $R^3LIVE$ -dataset (see Section 6.2.2). These two sequences were collected in front of a stairway with the LiDAR occasionally facing against the ground and a side wall (see Fig. 9(a) and (b)). When facing a wall, the LiDAR only observes a single plane, which is insufficient to determine the LiDAR pose, causing

TABLE 2 – Overview of the R<sup>3</sup>LIVE-dataset.

Sequence	Duration (s)	Traveling Length (m)	Sensor Degeneration	Return to origin <sup>1</sup>	Aruco marker <sup>2</sup>	Camera exposure time <sup>3</sup>	Scenarios
degenerate_seq_00	101	74.9	Camera, LiDAR	✓			Indoor
degenerate_seq_01	86	53.3	LiDAR	✓			Outdoor
degenerate_seq_02	85	75.2	LiDAR	✓			Outdoor
hku_campus_seq_00	202	190.6	—	✓			Indoor
hku_campus_seq_01	304	374.6	—				Outdoor
hku_campus_seq_02	323	354.3	—	✓		✓	Indoor, Outdoor
hku_campus_seq_03	173	181.2	—	✓		✓	Indoor, Outdoor
hku_main_building	1170	1036.9	—	✓		✓	Indoor, Outdoor
hku_park_00	351	401.8	—	✓			Outdoor, Cluttered
hku_park_01	228	247.3	—	✓	✓		Outdoor, Cluttered
hkust_campus_00	1073	1317.2	—	✓	✓		Indoor, Outdoor
hkust_campus_01	1162	1524.3	—	✓	✓		Indoor, Outdoor
hkust_campus_02	478	503.8	—	✓		✓	Indoor, Outdoor
hkust_campus_03	1618	2112.2	—			✓	Outdoor
<b>Total</b>	<b>7354</b>	<b>8447.6</b>					

<sup>1</sup> Sequences are collected by traveling a loop, with starting from and ending with the same position.

<sup>2</sup> Sequences with ArUco marker for providing the ground-truth relative pose.

<sup>3</sup> Sequences with ground-truth camera exposure time read from camera's API.

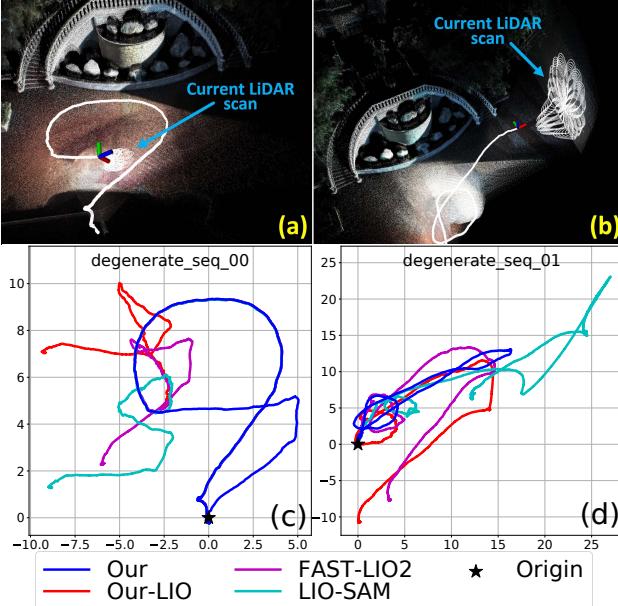


Fig. 9 – Tests in LiDAR degenerated environments.

LiDAR degeneration. The device starts from and ends at the same location, enabling the evaluation of localization drift. The estimated trajectories of our proposed system, our LIO-subsystem *Our-LIO*, and another two LIO systems, *FAST-LIO2* and *LIO-SAM*, are shown in Fig. 9(c) and (d). As can be seen, due to the LiDAR degeneration when facing a single plane, all three LiDAR-inertial odometry systems failed and did not return to the starting point. In contrast, by exploiting clues from the visual images, our proposed system works well in these two sequences and successfully returned to the starting point with drift down to 4.1 cm and 4.6 cm on sequences “degenerate\_seq\_00” and “degenerate\_seq\_01”, respectively. To obtain a more intuitive comprehension of the LiDAR degenerated scenarios, we recommend our readers to watch the accompanying video on YouTube: [youtu.be/qXrnIfn-7yA?t=390](https://youtu.be/qXrnIfn-7yA?t=390).

### 6.5.2 Evaluate of robustness in simultaneously LiDAR degenerated and visual texture-less environments

In this experiment, we challenge one of the most difficult scenarios in SLAM, where both LiDAR and camera

degenerate. We use sequence “degenerate\_seq\_02” of the R<sup>3</sup>LIVE-dataset, where the sensor device passes through a narrow “T”-shape passage (see Fig. 11) while occasionally facing against the side walls, causing LiDAR degeneration. Moreover, the visual texture on the white walls is very limited (Fig. 11(a) and Fig. 11(c)), especially for the wall-1, which has only changes in illumination. The absence of available LiDAR and visual features makes such scenarios rather challenging for both LiDAR-based and visual-based SLAM methods.

Taking advantage of the raw pixel color information and tightly fusing it with the LiDAR point cloud measurements, our proposed algorithm can “survive” in such extremely difficult scenarios. Fig. 12 shows our estimated pose, with the phases of passing through “wall-1” and “wall-2” shaded with purple and yellow, respectively. The estimated covariance is also shown in Fig. 12, which is bounded over the entire estimated trajectory, indicating that our estimation quality is stable over the entire process. The sensor is moved to the starting point, where an ArUco marker board is used to obtain the ground-true relative pose between the starting and end poses. Compared with the ground-true end pose, our algorithm drifts 1.62° in rotation and 4.57 cm in translation. We recommend the readers to the accompanying video on YouTube ([youtu.be/qXrnIfn-7yA?t=461](https://youtu.be/qXrnIfn-7yA?t=461)) for better visualization of the experiment.

### 6.6 Experiment-3: Evaluation of radiance map reconstruction

In this experiment, we evaluate the accuracy of our proposed algorithm in reconstructing the radiance map. Since the ground-true radiance map of the environment can not be measured, we evaluate the accuracy based on two indicators: one is the estimation quality of the camera exposure time and the other is the average photometric error between the reconstructed radiance map and the measured images.

TABLE 3 – The comparison of absolute position errors (APE, meters) on NCLT-dataset

Sequence (date)	Length (m)	Duration (hour:minute:second)	Our	R <sup>2</sup> LIVE	LVI-SAM	Our-LIO	Fast-LIO2	LIO-SAM
2012-01-08	6495.69	1 hr:25 min:35 sec	<b>10.81</b>	22.43	23.43	20.07	18.50	21.66
2012-01-15	7499.80	1 hr:52 min:19 sec	6.64	5.10	—	6.19	<b>4.81</b>	—
2012-01-22	6183.07	1 hr:27 min:22 sec	9.23	12.64	8.29	12.39	<b>7.14</b>	8.99
2012-02-02	6315.78	1 hr:38 min:36 sec	<b>5.33</b>	6.05	18.08	7.44	9.12	15.63
2012-02-04	5641.00	1 hr:18 min:30 sec	<b>5.58</b>	8.36	9.63	7.78	7.19	10.78
2012-02-05	6649.26	1 hr:34 min:17 sec	8.52	<b>7.58</b>	—	7.67	7.80	—
2012-02-12	5829.12	1 hr:25 min:35 sec	<b>4.50</b>	6.47	40.01	10.48	8.30	45.02
2012-02-18	6249.20	1 hr:29 min:55 sec	<b>40.50</b>	59.25	—	53.53	56.97	—
2012-02-19	6232.68	1 hr:29 min:11 sec	8.52	6.58	8.87	6.19	<b>5.98</b>	9.63
2012-03-17	5907.19	1 hr:22 min:53 sec	4.83	5.94	12.58	6.77	<b>4.70</b>	11.82
2012-03-31	6073.71	1 hr:27 min:53 sec	<b>4.94</b>	9.96	19.04	9.33	7.27	18.34
2012-04-29	3183.09	43 min:18 sec	6.32	6.43	5.94	6.27	6.44	<b>5.67</b>
2012-05-11	6116.74	1 hr:25 min:5 sec	<b>3.70</b>	3.79	4.23	4.21	4.13	4.16
2012-05-26	6340.70	1 hr:28 min:34 sec	<b>4.55</b>	6.30	18.34	6.13	6.43	18.38
2012-06-15	4085.89	55 min:10 sec	7.74	6.29	—	5.65	<b>5.27</b>	—
2012-08-04	5492.13	1 hr:20 min:32 sec	3.85	<b>3.73</b>	11.00	4.53	6.96	12.73
2012-08-20	6014.51	1 hr:23 min:48 sec	4.50	4.46	11.20	<b>4.18</b>	6.02	11.43
2012-09-28	5574.41	1 hr: 17 min:59 sec	7.89	<b>6.59</b>	34.42	6.84	10.42	36.71
2012-10-28	5682.10	1 hr:26 min:10 sec	7.71	7.95	—	8.61	<b>7.68</b>	—
2012-11-04	4788.33	1 hr:20 min:39 sec	7.48	9.31	3.42	12.55	<b>3.33</b>	3.37
2012-11-17	5751.89	1 hr:29 min:44 sec	8.68	6.48	21.92	6.13	<b>5.83</b>	24.17
2012-12-01	4991.93	1 hr:16 min:48 sec	11.25	14.21	6.93	16.96	<b>7.41</b>	7.21
2013-01-10	1137.32	17 min:4 sec	<b>3.41</b>	4.57	4.88	5.30	3.50	5.08
2013-02-23	5235.27	1hr: 20min:08 sec	<b>11.64</b>	13.39	12.60	13.79	11.85	12.20
2013-04-05	4523.65	1 hr:9 min:27 sec	8.82	11.91	9.83	11.55	<b>6.38</b>	9.01
<b>Total:</b>	137994.46	33 hr: 34 min: 52 sec						
<b>Average</b>			<b>8.51</b>	10.58	15.03	10.75	9.59	15.39

<sup>1</sup> Some systems fail in midway in some sequences and are marked as “—”.

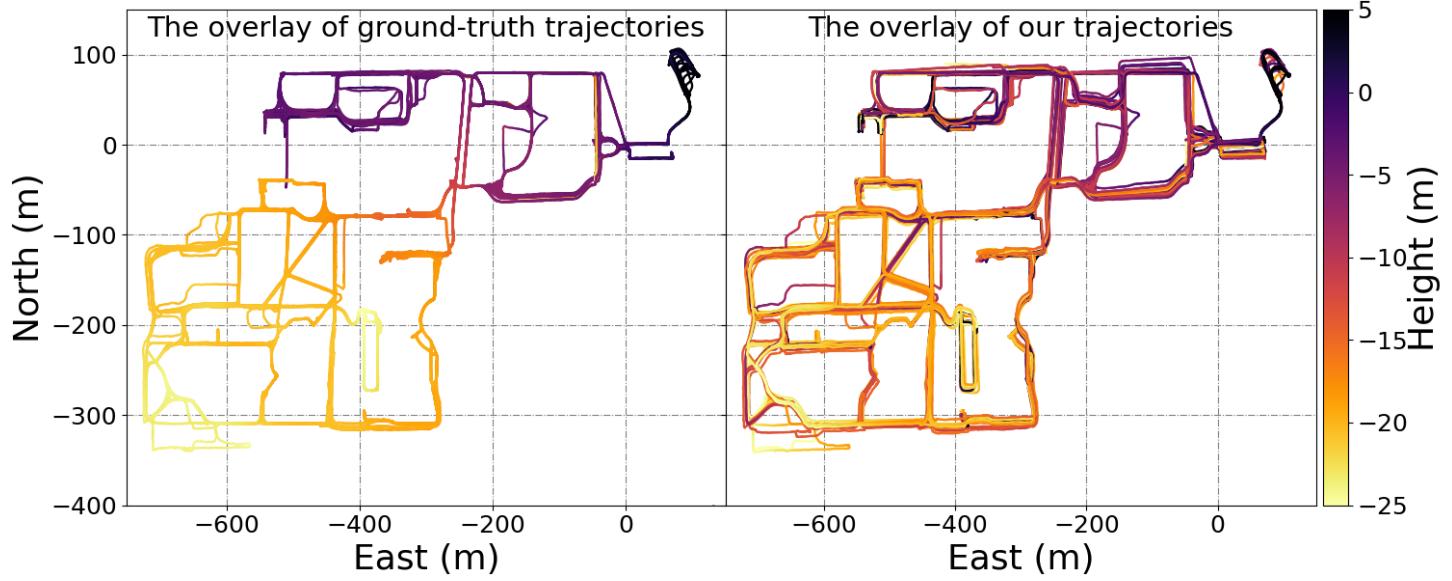


Fig. 10 – The overlay of ground-true trajectory and ours on NCLT-dataset.

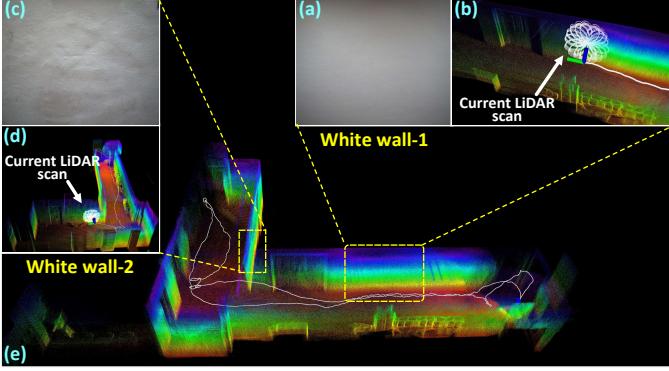


Fig. 11 – Tests in simultaneously LiDAR degenerated and visual texture-less environments.

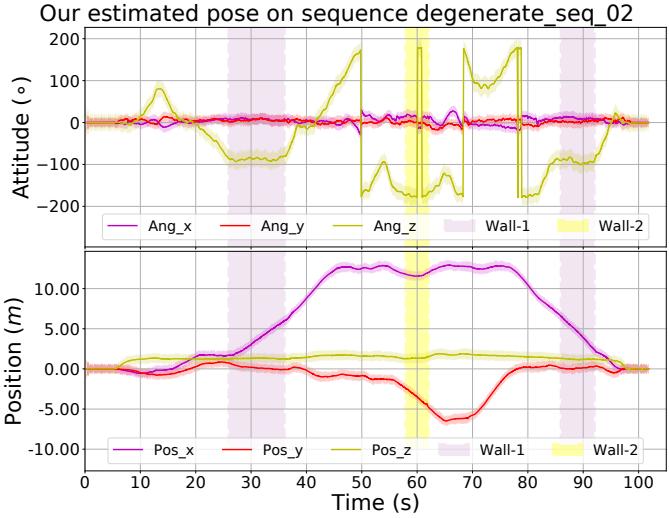


Fig. 12 – The estimated poses and their  $3\sigma$  bound with 5 times amplification for better visualization (the light-colored area around the trajectory) of the test in simultaneously LiDAR degenerated and visual texture-less environments. The shaded areas in purple and yellow are the phases of the sensors facing against the white “wall-1” and “wall-2”, respectively.

#### 6.6.1 Evaluation of exposure time estimation

In this experiment, we evaluate the accuracy of the estimated camera exposure time by comparing it with the ground-truth value read from the camera’s API. We use four sequences (see Fig. 13) of the R<sup>3</sup>LIVE-dataset, where the data were collected by traveling through both interior and exterior of a complex building to ensure significant changes in lighting conditions. We compare our estimated results with *Tum-cali* [44], which is currently the only work that can estimate the camera’s exposure time online to our knowledge. Both our system and [44] are initialized by assuming the exposure time of the first image frame is at a default value 1 ms.

The results are shown in Fig. 13, where the estimated exposure time of both our method and *Tum-cali* [44] is re-scaled to match with the ground-truth for better visualization. The average and maximum error of estimated exposure time w.r.t. the ground-truth is listed in Table 4. As shown in Fig. 13 and Table 4, our proposed method shows significantly lower estimation error than [44]. This is mainly because our method estimates the exposure time by minimizing the scan-to-map radiance error, while [44] recovers the exposure times from consecutive frames. The

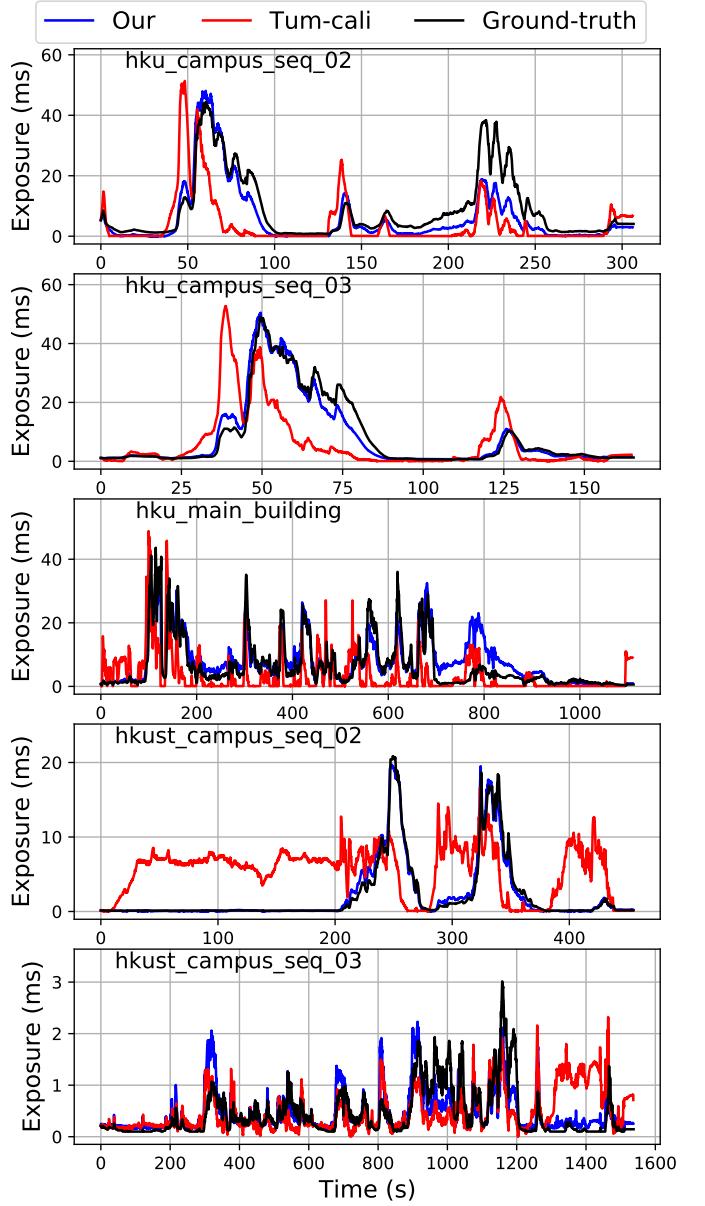


Fig. 13 – Estimation of camera exposure time.

TABLE 4 – The comparison of estimation error of exposure time over 5 sequences.

Sequence	Our Mean / Max (ms)	Tum_cali Mean / Max (ms)
hku_campus_seq_02	3.460 / 20.311	7.082 / 36.175
hku_campus_seq_03	1.460 / 10.653	6.400 / 37.126
hku_main_building	2.572 / 16.855	5.196 / 26.775
hkust_campus_seq_02	0.302 / 3.514	5.225 / 13.361
hkust_campus_seq_03	0.189 / 1.185	0.341 / 1.451

consequence is that our method can better utilize longer-term temporal intensity changes to restrain the drift of the exposure time estimation.

#### 6.6.2 Evaluation of radiance map

In this experiment, we evaluate the accuracy of our proposed algorithm in reconstructing the radiance map. Currently, LiDAR point cloud colorization remains one of the most challenging problems in the field of 3D reconstruction.

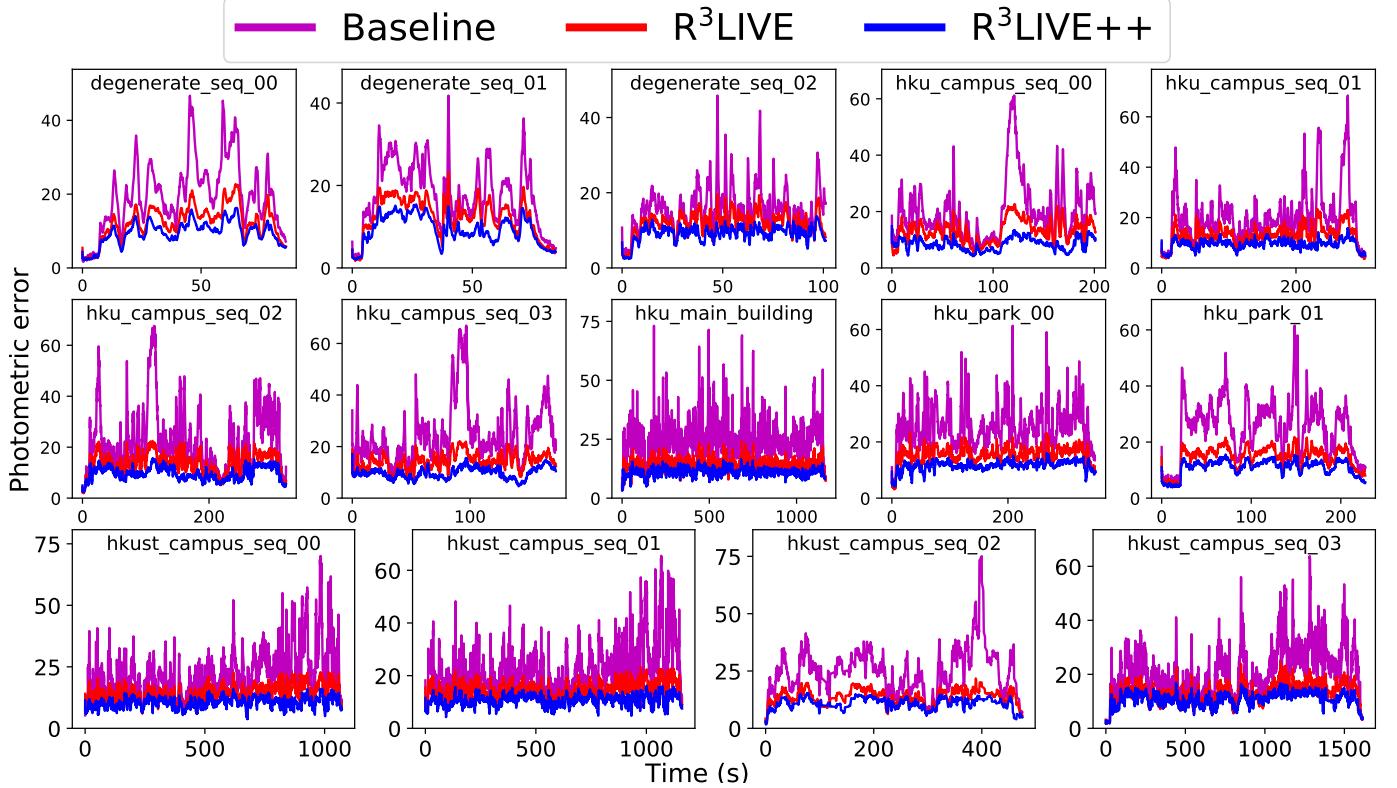


Fig. 14 – Photometric errors between the reconstructed radiance map and image pixels.



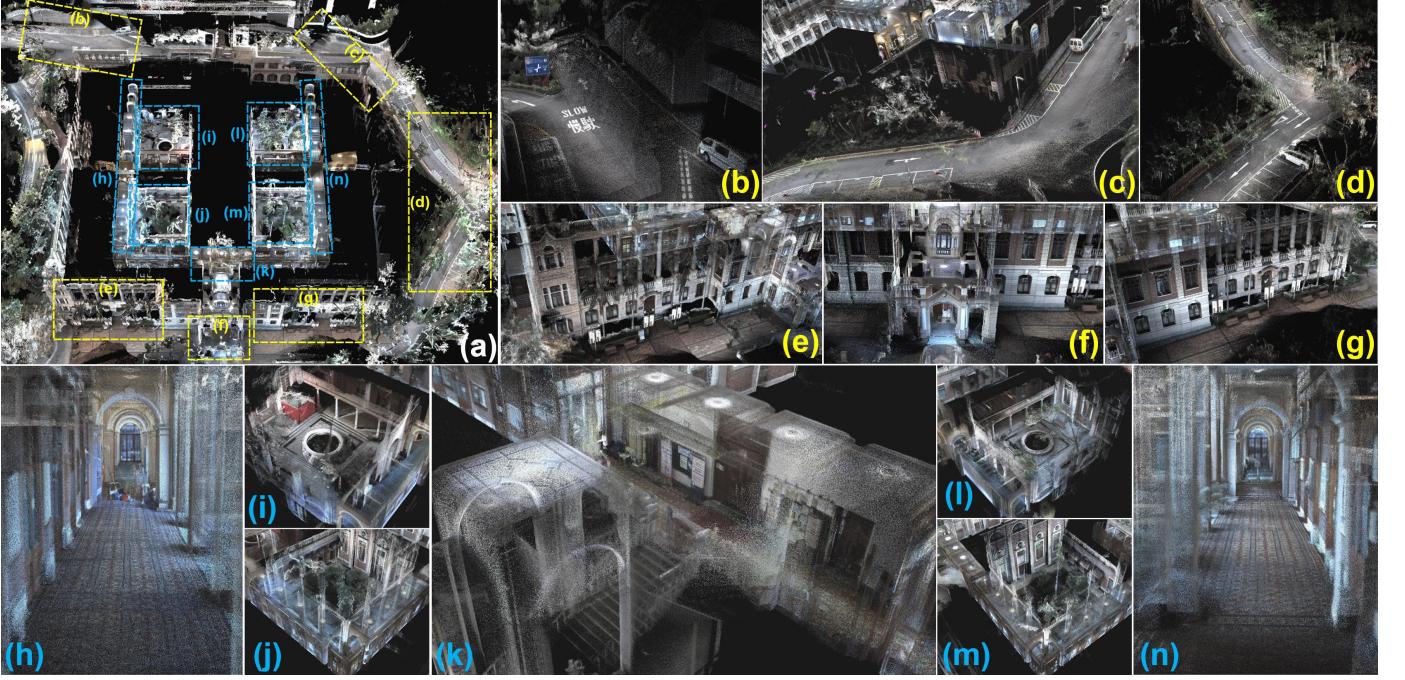
Fig. 15 – Closeup of a few scenes in the radiance map reconstructed by the baseline, R<sup>3</sup>LIVE and R<sup>3</sup>LIVE++.

TABLE 5 – The average photometric error among all sequences of R<sup>3</sup>LIVE-dataset.

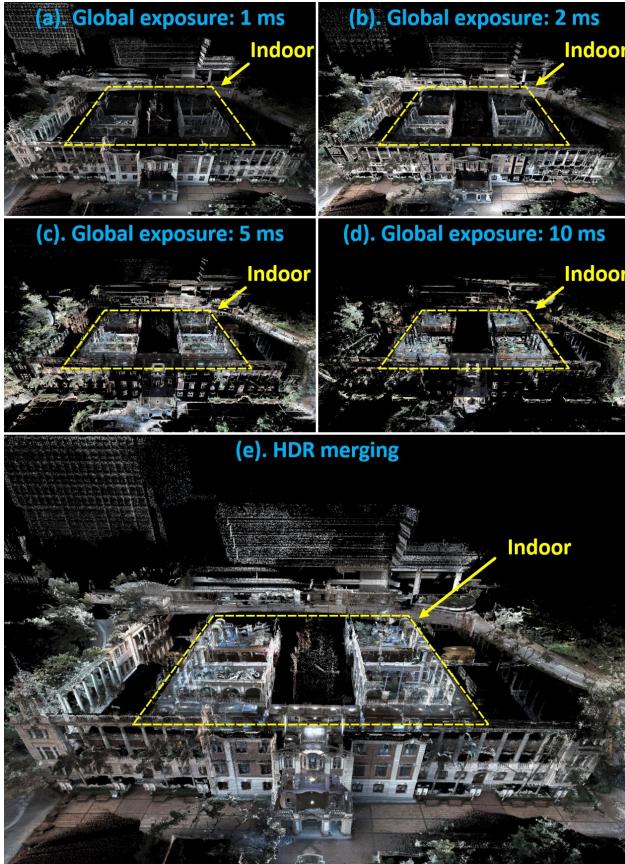
Sequence	Frames	baseline	R <sup>3</sup> LIVE	R <sup>3</sup> LIVE++
degenerate_seq_00	1694	30.58	21.36	16.19
degenerate_seq_01	1715	34.24	21.28	16.55
degenerate_seq_02	3315	27.14	20.30	15.97
hku_campus_seq_00	3016	34.78	22.56	14.57
hku_campus_seq_01	4502	34.97	22.47	16.30
hku_campus_seq_02	2595	39.01	23.73	16.85
hku_campus_seq_03	4845	42.95	24.78	16.93
hku_main_building	12157	43.12	22.26	19.04
hku_park_00	5251	43.86	27.01	20.07
hku_park_01	3410	43.29	25.17	19.02
hkust_campus_00	16075	37.64	24.19	18.00
hkust_campus_01	17426	39.09	24.67	18.53
hkust_campus_02	4031	41.74	23.77	18.92
hkust_campus_03	24270	35.60	22.37	18.65
<b>Average</b>	7413.57	38.60	23.58	18.01

The most common way is using the most recent image frame in time to give the color of each LiDAR frame [52], [53]. The preliminary implementation of our system R<sup>3</sup>LIVE published previously [42] colorized the point cloud by minimizing the photometric error similar to our current system but does not consider any exposure time estimation or photometric calibration. In this experiment, we compare our system against the previous implementation R<sup>3</sup>LIVE [42] to show the effectiveness of the exposure time estimation and photometric calibration and against the current baseline [52], [53] to show the advantage of the overall system.

To assess the radiance reconstruction error, after the map reconstruction, we re-project all points in the map



**Fig. 16** – Our reconstructed radiance map of the main building of HKU. (a) The bird’s view of the map, with its details shown in (b~n). (b~g) closeup of outdoor scenarios and (h~n) closeup of indoor scenarios. To see the real-time reconstruction process of the map, please refer to the video on YouTube: [youtu.be/qXrnIfn-7yA?t=55](https://youtu.be/qXrnIfn-7yA?t=55).



**Fig. 17** – (a~ d) Images rendered from the reconstructed radiance map at different exposure time: 1 ms, 2 ms, 5 ms and 10 ms. (e) The HDR image merged from (a~d). Notice that for the sake of better visualization, those points that are over-exposure are not displayed.

with radiance information to each image frame with the estimated camera pose and calibrated photometric parameters. Then, we calculate the photometric error between

the map point color and the RGB values of the image at the projected pixel location. The average photometric error of each image frame is calculated to evaluate the accuracy of the reconstructed radiance map. We perform the evaluation on all R<sup>3</sup>LIVE-dataset sequences, with the results of each sequence are given in Fig. 14, and the average photometric of each sequence is listed in Table 5. As can be seen, our system has consistently achieved the lowest photometric errors in all sequences and the next best is R<sup>3</sup>LIVE. Fig. 15 shows a few closeups of the reconstructed radiance map, from which we can clearly tell the words on objects. Moreover, in Fig. 16, we present the reconstructed radiance map of the sequence “hku\_main\_building” in R<sup>3</sup>LIVE-dataset, in which we collect the data in both interior and exterior of the main building of HKU. As shown in Fig. 16, both the indoor and outdoor details (e.g., marks on the road) are very clear, demonstrating that our proposed algorithm is of high accuracy. What worth mentioning is, this 3D radiance map is reconstructed on the fly as the data is being acquired (see the accompanying video on YouTube: [youtu.be/qXrnIfn-7yA?t=55](https://youtu.be/qXrnIfn-7yA?t=55)). For more qualitative results of other sequences, we refer our readers to our Supplementary Material: [github.com/hku-mars/r3live/blob/master/supply/r3live\\_plus\\_plus\\_supplementary\\_material.pdf](https://github.com/hku-mars/r3live/blob/master/supply/r3live_plus_plus_supplementary_material.pdf)

## 6.7 Run time analysis

In this section, We investigate the average time consumption of our proposed system on a CPU-Only PC (equipped with an Intel i7-9700K CPU and 64 GB RAM). We counts the average time consumption on all sequences of both datasets (i.e., the NCLT-dataset and R<sup>3</sup>LIVE-dataset), whose results are shown in Table 6 and Table 7, respectively. For the NCLT-dataset, each LiDAR scan takes an average of 34.3 ms and each camera image takes an average of 16.6 ms processing

**TABLE 6** – The average time consumption per LiDAR or camera frame of our system on NCLT-dataset.

NCLT-dataset Sequence (date)	LiDAR frame Mean / Std (ms)	Camera frame Mean / Std (ms)
2012-01-08	33.852 / 9.110	15.911 / 4.180
2012-01-15	35.517 / 11.602	17.613 / 4.275
2012-01-22	34.392 / 11.719	16.551 / 4.591
2012-02-02	33.812 / 11.188	16.926 / 4.361
2012-02-04	32.599 / 10.498	15.09 / 4.164
2012-02-05	34.823 / 11.147	17.09 / 4.276
2012-02-12	32.738 / 11.765	17.071 / 3.846
2012-02-18	37.284 / 10.169	17.087 / 3.973
2012-02-19	38.004 / 9.928	18.179 / 4.099
2012-03-17	32.196 / 11.154	17.304 / 4.370
2012-03-31	36.283 / 10.377	16.228 / 4.312
2012-04-29	33.652 / 9.014	16.487 / 4.018
2012-05-11	34.044 / 8.964	17.357 / 3.811
2012-05-26	37.623 / 11.315	15.228 / 4.384
2012-06-15	29.07 / 8.807	17.254 / 3.881
2012-08-04	29.231 / 10.427	17.267 / 4.629
2012-08-20	28.561 / 10.952	15.73 / 3.931
2012-09-28	36.692 / 10.738	15.458 / 4.346
2012-10-28	36.147 / 11.497	16.081 / 4.102
2012-11-04	37.066 / 11.430	16.242 / 4.061
2012-11-17	38.153 / 9.187	16.931 / 3.904
2012-12-01	36.357 / 11.660	16.492 / 4.165
2013-01-10	32.977 / 10.355	18.328 / 3.971
2013-02-23	36.171 / 10.720	15.368 / 4.358
2013-04-05	29.542 / 10.052	15.725 / 4.189
<b>Average</b>	<b>34.271 / 10.551</b>	<b>16.600 / 4.168</b>

**TABLE 7** – The average time consumption per LiDAR or camera frame of our system on R<sup>3</sup>LIVE-dataset.

Sequence (date)	LiDAR frame Mean / Std (ms)	Camera frame Mean / Std (ms)
degenerate_seq_00	17.927 / 9.080	13.057 / 2.675
degenerate_seq_01	8.111 / 3.622	12.111 / 2.502
degenerate_seq_02	21.638 / 6.793	13.543 / 3.014
hku_campus_seq_00	27.659 / 8.159	18.829 / 3.719
hku_campus_seq_01	25.328 / 11.246	18.187 / 2.860
hku_campus_seq_02	20.757 / 5.109	16.928 / 2.518
hku_campus_seq_03	21.705 / 4.903	16.040 / 2.338
hku_main_building	25.123 / 10.246	17.023 / 2.512
hku_park_00	26.908 / 7.196	17.882 / 2.545
hku_park_01	24.412 / 6.598	17.479 / 2.485
hkust_campus_seq_00	28.160 / 9.703	16.624 / 2.756
hkust_campus_seq_01	27.058 / 10.135	16.138 / 2.813
hkust_campus_seq_02	22.857 / 6.135	16.518 / 2.618
hkust_campus_seq_03	30.757 / 5.109	16.928 / 2.518
<b>Average</b>	<b>23.453 / 7.431</b>	<b>16.234 / 2.705</b>

time. Since the data rate of the LiDAR and camera sensors are 10 Hz and 5 Hz, the total processing time per second is 426 ms, comprising of the time for processing 10 lidar scans (i.e., 343 ms) and 5 images (83 ms). For R<sup>3</sup>LIVE-dataset, each LiDAR scan takes an average of 22.7 ms and each camera image takes an average of 16.2 ms processing time. Since the data rate of the LiDAR and camera sensors are 10 Hz and 15 Hz, the total processing time per second is 470 ms, comprising of the time for processing 10 lidar scans (i.e., 235 ms) and 15 images (244 ms). In both cases, the processing time required per second is below half a second, indicating that our system runs in real-time (even two times faster than real-time) in both pose estimation and radiance map reconstruction. In addition, the processing time per image is similar across the two datasets while that for each lidar scan is quite different. The reason is that the lidar in R<sup>3</sup>LIVE-dataset has a much lower data rate than that of the NCLT-dataset (240 k versus 695 k points per second) while

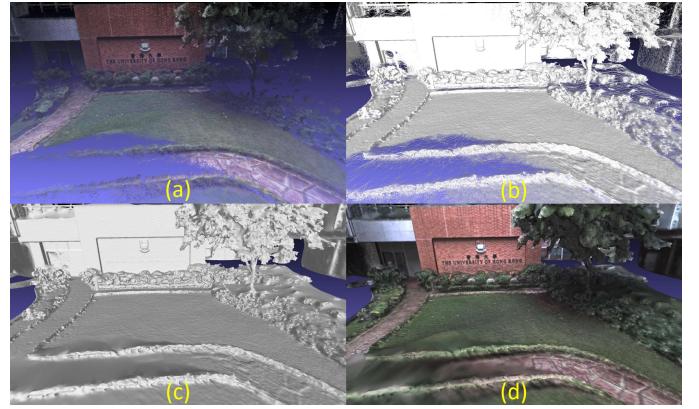
the cameras have similar resolution.

## 7 APPLICATIONS WITH R<sup>3</sup>LIVE

### 7.1 High dynamic range (HDR) imaging

After the reconstruction of the radiance map, we are able to render an image by projecting the map to an image plane with a given pose and exposure time with equation (3). Taking the sequence “hku\_main\_building” as an example, Fig. 17(a), (b), (c) and (d) are the rendered images with global exposure time of 1 ms, 2 ms, 5 ms and 10 ms, respectively. These images rendered at different exposure times can be merged into a HDR image shown in Fig. 17 (e).

### 7.2 Mesh reconstruction and texturing



**Fig. 18** – (a) show the RGB-colored 3D points reconstructed by R<sup>3</sup>LIVE++. (b) and (c) show the wireframe and surface of our reconstructed mesh. (d) show the mesh after texture rendering.

While R<sup>3</sup>LIVE++ reconstructs the colored 3D map in real-time, we also develop software utilities to mesh and texture the reconstructed map offline (see Fig. 18). For meshing, we make use of the Delaunay triangulation and graph cuts [54] implemented in CGAL [55] and openMVS [56]. After the mesh construction, we texture the mesh with the vertex (point) colors, with are rendered by our VIO subsystem.

Our developed utilities also export the colored point map from R<sup>3</sup>LIVE++ or the offline meshed map into commonly used file formats such as “pcd”, “ply”, “obj”, etc. As a result, the maps reconstructed by R<sup>3</sup>LIVE++ can be imported into various 3D software, including but not limited to CloudCompare [57], Meshlab [58], AutoDesks 3ds Max [59].

### 7.3 Toward various of 3D applications

With the developed software utilities, we can export the reconstructed 3D maps to Unreal Engine [60] to enable a series of 3D applications. For example, in Fig. 19(a), we built a car simulator with the AirSim [61]. In Fig. 19(b) and Fig. 19(c), we used our reconstructed maps to develop video games for mobile platforms and desktop PCs, respectively. To get more details about our demos, we refer the readers to watch our video on YouTube: [youtu.be/qXrnIfn-7yA?t=516](https://youtu.be/qXrnIfn-7yA?t=516).



**Fig. 19** – In (a), we built a car simulator with our maps and AirSim. The images in yellow and blue frame-boxes are the depth and RGB image query from the AirSim’s API. In (b) and (c), we developed video games for mobile platforms and desktop PCs. The player in (b) is controlling the actor to explore the campus of HKU, and in (c) is fighting against a dragon by shooting rubber balls at HKUST.

## 8 CONCLUSIONS AND FUTURE WORK

### 8.1 Conclusions

In this paper, we proposed a novel LiDAR-inertial-visual fusion framework termed R<sup>3</sup>LIVE++ to achieve robust and accurate state estimation while simultaneously reconstructing the radiance map on the fly. This framework consists of two subsystems (i.e., the LIO and the VIO) that jointly and incrementally build a 3D radiance map of the environment in real-time. By tightly fusing the measurement of three different types of sensors, R<sup>3</sup>LIVE++ can achieve higher localization accuracy while being robust enough to scenarios with sensor degenerations.

In our experiments, we extensively validated our proposed algorithm with real-world experiments in terms of localization accuracy, robustness, and radiance map reconstruction accuracy. The benchmark results on 25 sequences from an open dataset (the NCLT-dataset) showed that R<sup>3</sup>LIVE++ achieved the highest overall accuracy among all other state-of-the-art SLAM systems under comparison. The evaluation on a private dataset (the R<sup>3</sup>LIVE-dataset) showed that R<sup>3</sup>LIVE++ was robust to extremely challenging scenarios that LiDAR and/or camera measurements degenerate (e.g., when the device is facing a single texture-less wall). Finally, compared with other counterparts, R<sup>3</sup>LIVE++ estimates the camera exposure time more accurately and reconstructs the true radiance information of the environment with significantly smaller errors when compared to the measured values in images.

To demonstrate the extendability of our work, we developed several applications based on our reconstructed radiance maps, such as high dynamic range (HDR) imaging, virtual environment exploration, and 3D video gaming. Finally, to share our findings and make contributions to the community, we made our codes, hardware design, and dataset publicly available on our Github.

### 8.2 Future work

In R<sup>3</sup>LIVE++, the radiance map is reconstructed with 3D points that contain radiance information, which prevents us from rendering high-resolution images from the radiance map due to the limited point cloud density of the radiance map (1 cm in our current implementation). While further increasing this point cloud density is possible, it will further increase the processing time. This point density is also limited by the density of the raw points measured by lidar sensors. Noticing that images often have much higher resolution, in the future, we could explore how to make full use of such high-resolution images in the fusion framework.

## REFERENCES

- [1] F. Gao, W. Wu, W. Gao, and S. Shen, “Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments,” *Journal of Field Robotics*, vol. 36, no. 4, pp. 710–733, 2019.
- [2] F. Kong, W. Xu, Y. Cai, and F. Zhang, “Avoiding dynamic small obstacles with onboard sensing and computation on aerial robots,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7869–7876, 2021.
- [3] H. Lategahn, A. Geiger, and B. Kitt, “Visual slam for autonomous ground vehicles,” in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1732–1737.
- [4] P. Beinschob and C. Reinke, “Graph slam based mapping for agv localization in large-scale warehouses,” in *2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 2015, pp. 245–248.
- [5] J. Lin, X. Liu, and F. Zhang, “A decentralized framework for simultaneous calibration, localization and mapping with multiple lidars,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4870–4877.
- [6] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, et al., “Towards fully autonomous driving: Systems and algorithms,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 163–168.
- [7] G. Ros, A. Sappa, D. Ponsa, and A. M. Lopez, “Visual slam for driverless cars: A brief survey,” in *Intelligent Vehicles Symposium (IV) Workshops*, vol. 2, 2012, pp. 1–6.
- [8] A. Singandhupe and H. M. La, “A review of slam techniques and security in autonomous driving,” in *2019 third IEEE international conference on robotic computing (IRC)*. IEEE, 2019, pp. 602–607.
- [9] O. Pink, “Visual map matching and localization using a global feature map,” in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2008, pp. 1–7.
- [10] M. Bürki, M. Dymczyk, I. Gilitschenski, C. Cadena, R. Siegwart, and J. Nieto, “Map management for efficient long-term visual localization in outdoor environments,” in *2018 IEEE Intelligent vehicles symposium (IV)*. IEEE, 2018, pp. 682–688.
- [11] B. Nagy and C. Benedek, “Real-time point cloud alignment for vehicle localization in a high resolution 3d map,” in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.
- [12] W. Li, C. Pan, R. Zhang, J. Ren, Y. Ma, J. Fang, F. Yan, Q. Geng, X. Huang, H. Gong, et al., “Aads: Augmented autonomous driving simulation using data-driven algorithms,” *Science robotics*, vol. 4, no. 28, p. eaaw0863, 2019.
- [13] Z. Bao, S. Hossain, H. Lang, and X. Lin, “High-definition map generation technologies for autonomous driving: A review,” *arXiv preprint arXiv:2206.05400*, 2022.
- [14] J. Lin and F. Zhang, “Loam\_livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3126–3131.
- [15] J. Zhang and S. Singh, “Loam: Lidar odometry and mapping in real-time,” in *Robotics: Science and Systems*, vol. 2, no. 9, 2014.
- [16] T. Shan and B. Englot, “Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.
- [17] J. Lin and F. Zhang, “A fast, complete, point cloud based loop closure for lidar odometry and mapping,” *arXiv preprint arXiv:1909.11811*, 2019.
- [18] H. Ye, Y. Chen, and M. Liu, “Tightly coupled 3d lidar inertial odometry and mapping,” in *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.
- [19] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, “Liosam: Tightly-coupled lidar inertial odometry via smoothing and mapping,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5135–5142.
- [20] K. Li, M. Li, and U. D. Hanebeck, “Towards high-performance solid-state-lidar-inertial odometry and mapping,” *arXiv preprint arXiv:2010.13150*, 2020.
- [21] C. Qin, H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu, “Lins: A lidar-inertial state estimator for robust and efficient navigation,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8899–8906.

- [22] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter," *IEEE Robotics and Automation Letters*, 2021, in press.
- [23] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, pp. 1–21, 2022.
- [24] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.
- [25] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.
- [26] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 2007, pp. 225–234.
- [27] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.
- [28] R. Mur-Artal and J. D. Tardos, "Orb-slam2: An open-source slam system for monocular, stereo, and rgbd cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.
- [29] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.
- [30] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.
- [31] B. D. Lucas, T. Kanade, et al., "An iterative image registration technique with an application to stereo vision." Vancouver, 1981.
- [32] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*. Springer, 2014, pp. 834–849.
- [33] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 15–22.
- [34] J. Zhang and S. Singh, "Laser-visual-inertial odometry and mapping with high robustness and low drift," *Journal of Field Robotics*, vol. 35, no. 8, pp. 1242–1264, 2018.
- [35] W. Shao, S. Vijayarangan, C. Li, and G. Kantor, "Stereo visual inertial lidar simultaneous localization and mapping," *arXiv preprint arXiv:1902.10741*, 2019.
- [36] T. Laidlow, M. Bloesch, W. Li, and S. Leutenegger, "Dense rgbd-inertial slam with map deformations," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6741–6748.
- [37] Y. Zhu, C. Zheng, C. Yuan, X. Huang, and X. Hong, "Camvox: A low-cost and accurate lidar-assisted visual slam system," *arXiv preprint arXiv:2011.11357*, 2020.
- [38] X. Zuo, P. Geneva, W. Lee, Y. Liu, and G. Huang, "Lic-fusion: Lidar-inertial-camera odometry," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5848–5854.
- [39] X. Zuo, Y. Yang, J. Lv, Y. Liu, G. Huang, and M. Pollefeys, "Lic-fusion 2.0: Lidar-inertial-camera odometry with sliding-window plane-feature tracking," in *IROS 2020*, 2020.
- [40] T. Shan, B. Englot, C. Ratti, and D. Rus, "Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping," *arXiv preprint arXiv:2104.10831*, 2021.
- [41] J. Lin, C. Zheng, W. Xu, and F. Zhang, "R<sup>2</sup>live: A robust, real-time, lidar-inertial-visual tightly-coupled state estimator and mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7469–7476, 2021.
- [42] J. Lin and F. Zhang, "R<sup>3</sup>live: A robust, real-time, rgbd-colored, lidar-inertial-visual tightly-coupled state estimation and mapping package," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10672–10678.
- [43] D. He, W. Xu, and F. Zhang, "Embedding manifold structures into kalman filters," *arXiv preprint arXiv:2102.03804*, 2021.
- [44] P. Bergmann, R. Wang, and D. Cremers, "Online photometric calibration of auto exposure video for realtime visual odometry and slam," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 627–634, 2017.
- [45] J. Engel, V. Usenko, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry," *arXiv preprint arXiv:1607.02555*, 2016.
- [46] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.
- [47] R. Wang, M. Schworer, and D. Cremers, "Stereo dso: Large-scale direct sparse visual odometry with stereo cameras," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3903–3911.
- [48] T. Qin and S. Shen, "Online temporal calibration for monocular visual-inertial systems," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3662–3669.
- [49] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of michigan north campus long-term vision and lidar dataset," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2016.
- [50] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.
- [51] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7244–7251.
- [52] Y. Zhu, C. Zheng, C. Yuan, X. Huang, and X. Hong, "Camvox: A low-cost and accurate lidar-assisted visual slam system," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5049–5055.
- [53] C. Zheng, Q. Zhu, W. Xu, X. Liu, Q. Guo, and F. Zhang, "Fastlivo: Fast and tightly-coupled sparse-direct lidar-inertial-visual odometry," *arXiv preprint arXiv:2203.00893*, 2022.
- [54] P. Labatut, J.-P. Pons, and R. Keriven, "Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts," in *2007 IEEE 11th international conference on computer vision*. IEEE, 2007, pp. 1–8.
- [55] A. Fabri and S. Pion, "Cgal: The computational geometry algorithms library," in *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2009, pp. 538–539. [Online]. Available: <https://www.cgal.org>
- [56] D. Cernea, "OpenMVS: Multi-view stereo reconstruction library," 2020. [Online]. Available: <https://cdcseacave.github.io/openMVS>
- [57] D. Girardeau-Montaut, "Cloudcompare," France: EDF R&D Telecom ParisTech, 2016. [Online]. Available: <https://www.danielgm.net/cc>
- [58] P. Cignoni, G. Ranzuglia, M. Callieri, M. Corsini, F. Ganovelli, N. Pietroni, and M. Tarini, "Meshlab," 2011. [Online]. Available: <https://www.meshlab.net>
- [59] Autodesk, "Autodesk 3ds max." [Online]. Available: <https://www.autodesk.com/products/3ds-max>
- [60] Epic Games, "Unreal engine." [Online]. Available: <https://www.unrealengine.com>
- [61] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*. Springer, 2018, pp. 621–635. [Online]. Available: <https://microsoft.github.io/AirSim>



**Jiarong Lin** received a B.S. degree in Optical Information Science and Technology from the University of Electronic Science and Technology of China (UESTC) in 2015. He is currently a Ph.D. candidate in the Department of Mechanical Engineering, the University of Hong Kong (HKU), Hong Kong, China.

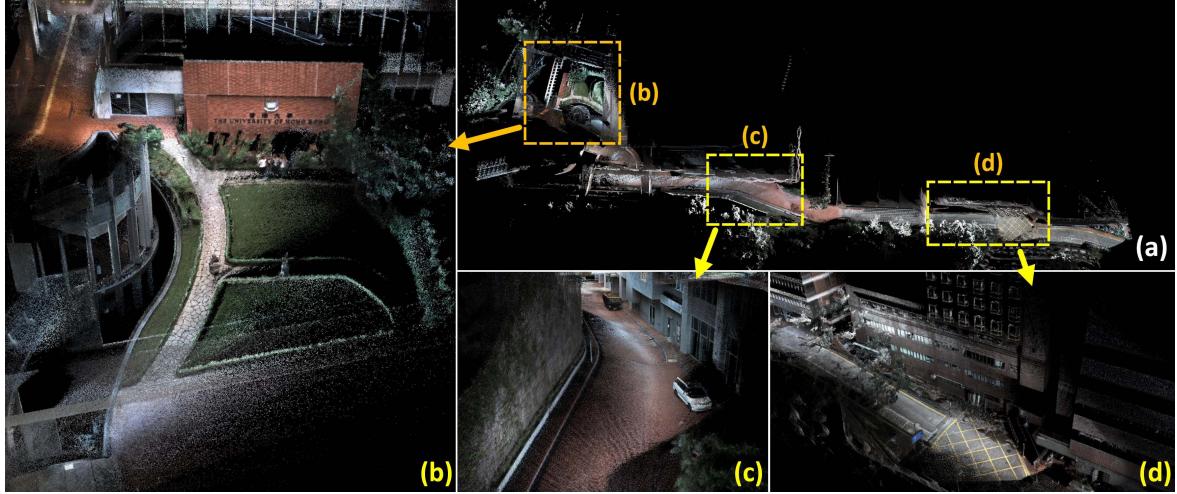
His research interests include light detection and ranging (LiDAR) mapping and sensor fusion.



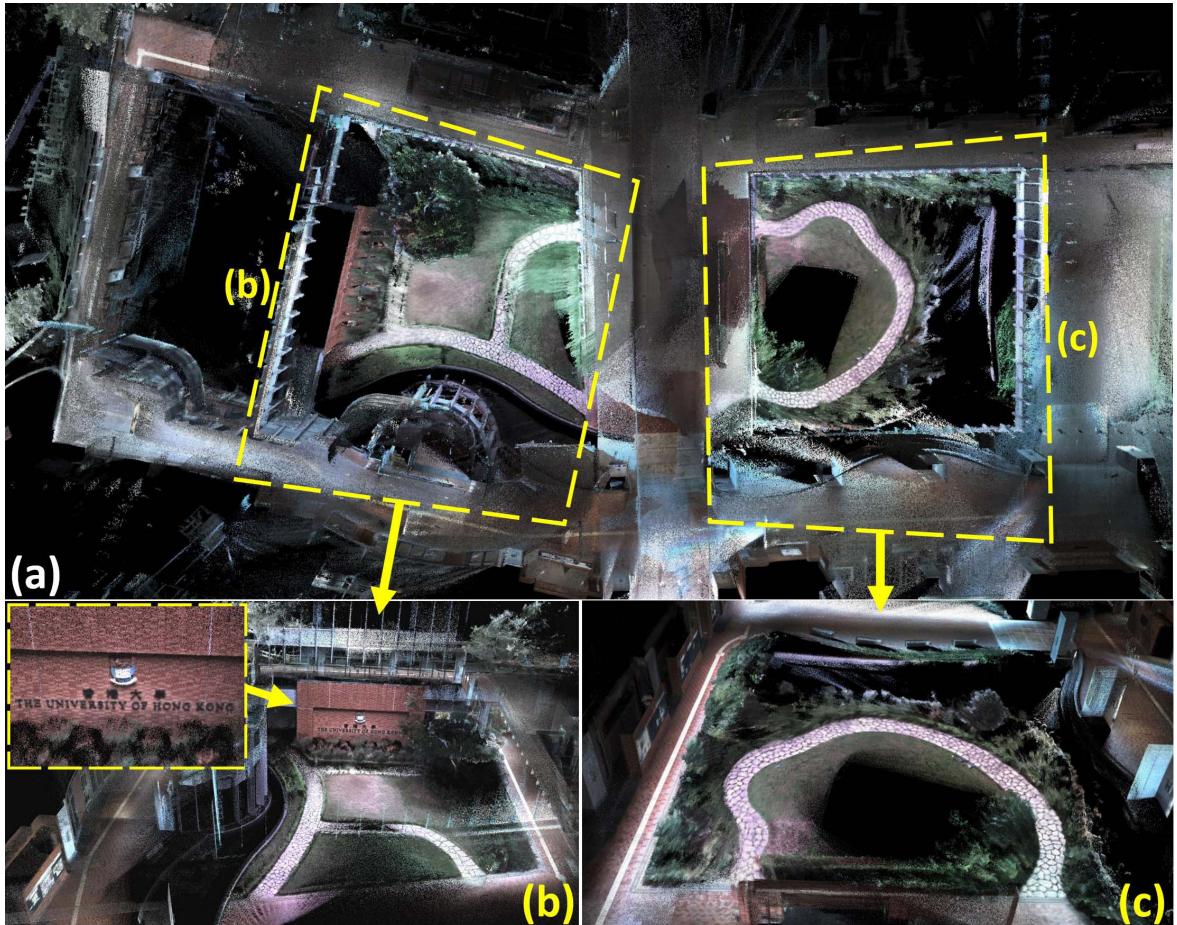
**Fu Zhang** received the B.E. degree in automation from the University of Science and Technology of China (USTC), Hefei, Anhui, China, in 2011, and the Ph.D. degree in controls from the University of California at Berkeley, Berkeley, CA, USA, in 2015.

He joined the Department of Mechanical Engineering, The University of Hong Kong (HKU), Hong Kong, as an Assistant Professor in August 2018. His current research interests are on robotics and controls, with a focus on unmanned aerial vehicle (UAV) design, navigation, control, and light detection and ranging (LiDAR)-based simultaneous localization and mapping (SLAM).

## Supplementary Material for R<sup>3</sup>LIVE++: The qualitative results of our reconstructed radiance map on R<sup>3</sup>LIVE-dataset



**Fig. 1** – Sequence “hku\_campus\_seq\_01” are collected by walking along the drive way of the HKU campus. (a) is the birdview of the whole radiance map, with its details shown in (b~ d).



**Fig. 2** – Sequence “hku\_campus\_seq\_00/02/03” are sampled at the same place but at different times of day (evening, noon and morning, respectively) and with different traveling trajectories. (a) is the birdview of map of sequence “hku\_campus\_seq\_02”, with the closeup view of details are shown in (b) and (c).

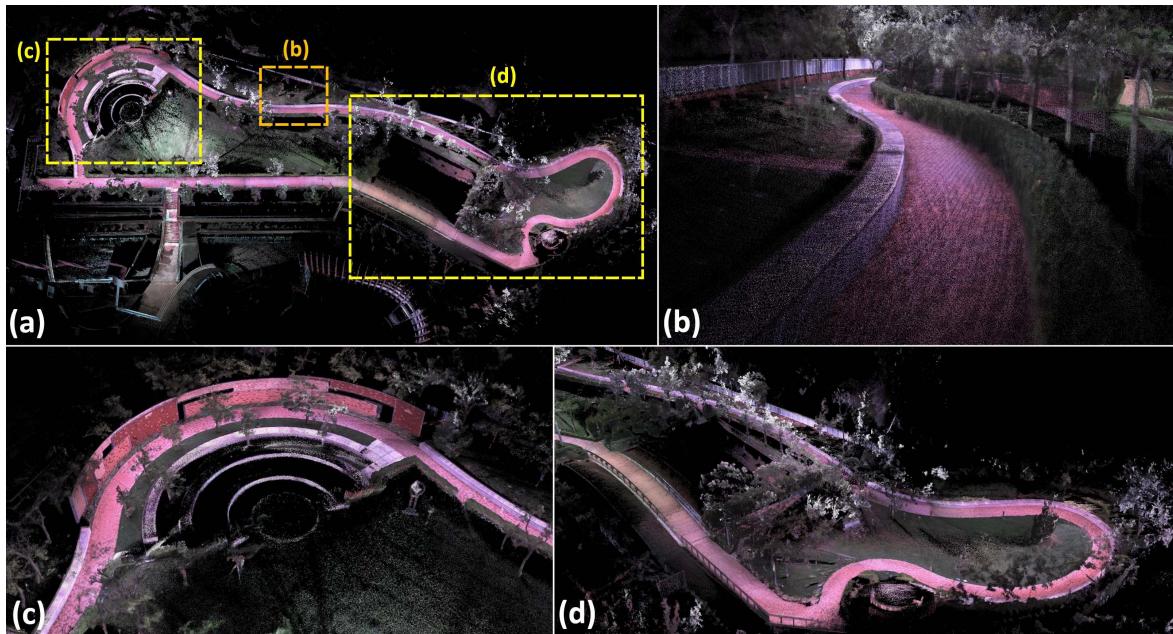


Fig. 3 – Sequence “hku\_park\_00” is collected by walking along the pathway of a garden of HKU. (a) is the birdview of the whole radiance map, with its details shown in (b~ d).

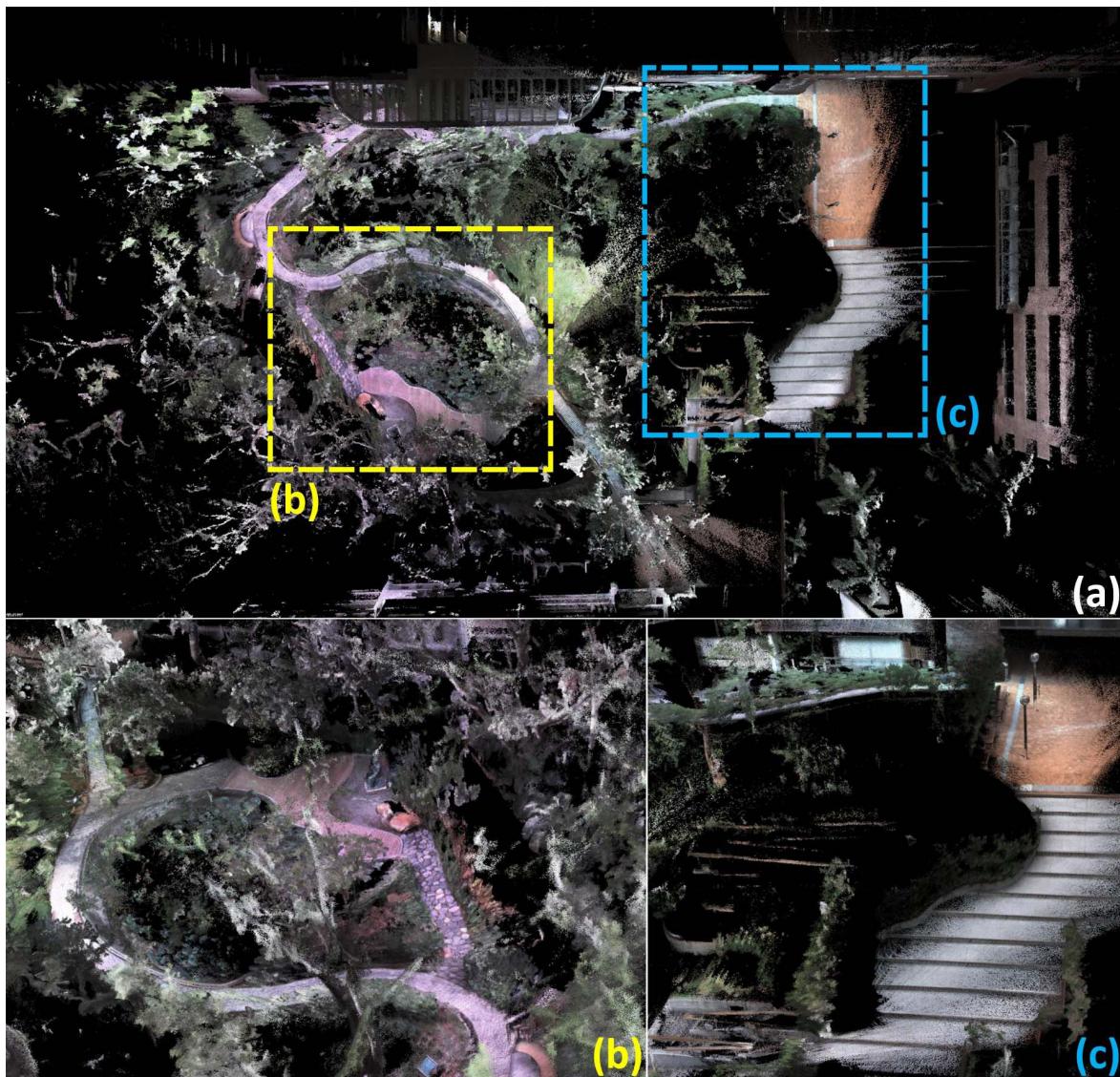


Fig. 4 – Sequence “hku\_park\_01” is collected in a cluttered environment with many trees and bushes. (a) is the birdview of the whole radiance map, with its details are shown in (b) and (c).

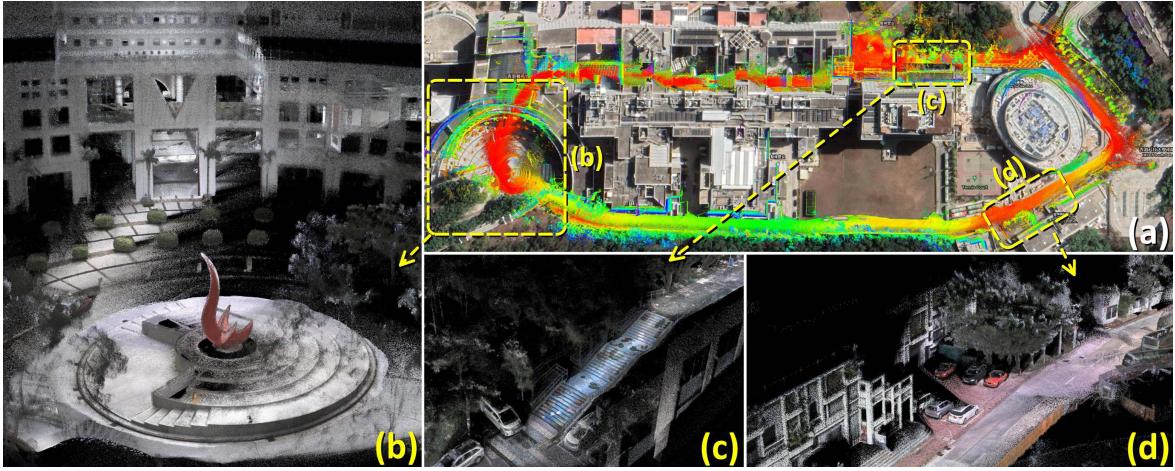


Fig. 5 – Sequence “hku\_campus\_seq\_00/01” are collected within the campus of HKUST with two different traveling trajectories. In (a), we merge the point cloud of sequence “hku\_campus\_seq\_00” with the GoogleEarth satellite image and find them aligned well. The details of our reconstructed radiance map are selectively shown in (b~d).

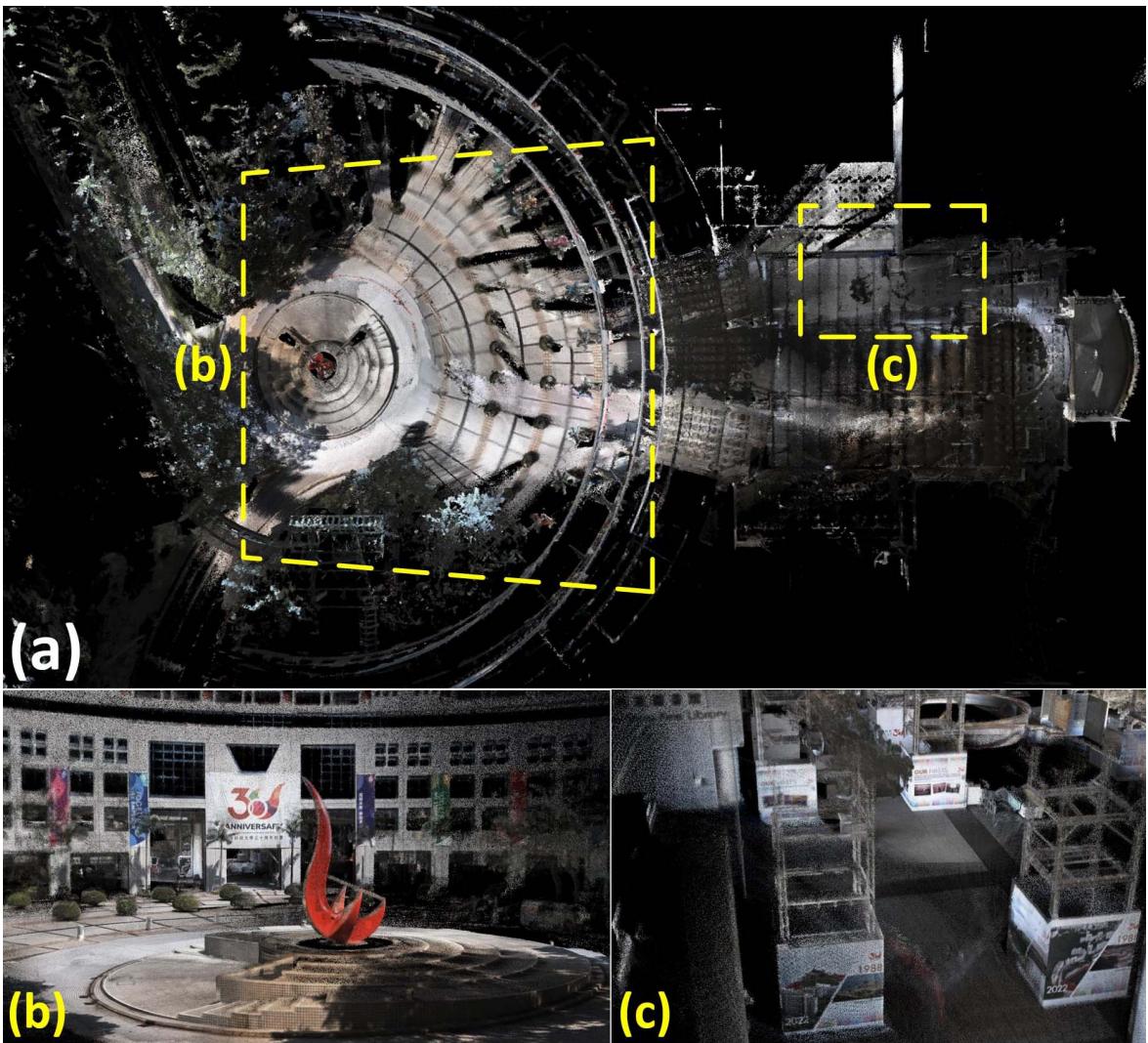
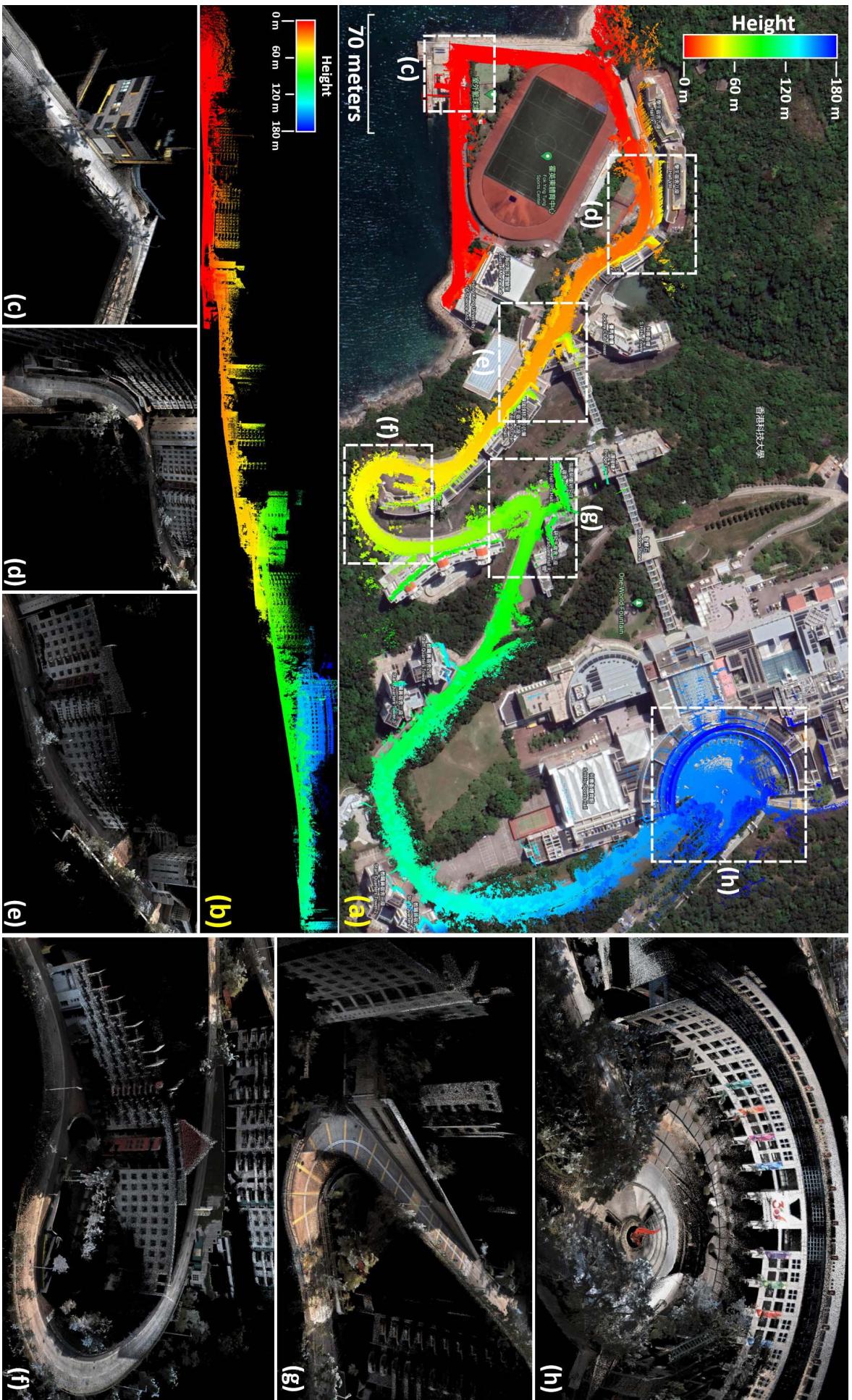


Fig. 6 – Sequence “hku\_campus\_seq\_02” is collected by exploring the entrance piazza of HKUST, traveling both the interior and exterior of the buildings. (a) is the birdview of the whole radiance map, with the outdoor and indoor scenarios selectively shown in (b) and (c), respectively.



**Fig. 7 – Sequence “hkust\_campus\_seq\_03”** captures most part of the HKUST’s campus, with the traveling length reaching 2.1 km. We collected the data starting from the sea front (see the lower left of (a)) and ending at the entrance piazza (the upper right of (a)) of HKUST. In (a), we merge our reconstructed point cloud map (points are colored by their height) with the Google Earth satellite image and find them aligned well. (b) shows the side view of the map. (c~h) are the closeup views of the details marked in (a). To see the real-time reconstruction process of the map, please refer to the video on YouTube: [youtu.be/qXrnIfn-7ya?t=261](https://youtu.be/qXrnIfn-7ya?t=261).