PL-CVIO: Point-Line Cooperative Visual-Inertial Odometry

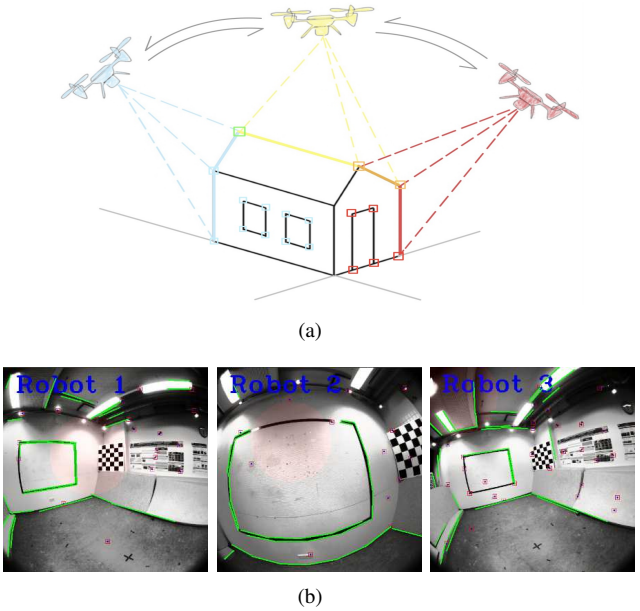
Yanyu Zhang, Pengxiang Zhu, and Wei Ren

***Abstract*— Low-feature environments are one of the main Achilles' heels of geometric computer vision (CV) algorithms. In most human-built scenes often with low features, lines can be considered complements to points. In this paper, we present a multi-robot cooperative visual-inertial navigation system (VINS) using both point and line features. By utilizing the *covariance intersection* (CI) update within the multi-state constraint Kalman filter (MSCKF) framework, each robot exploits not only its own point and line measurements, but also constraints of common point and common line features observed by its neighbors. The line features are parameterized and updated by utilizing the *Closest Point* representation. The proposed algorithm is validated extensively in both Monte-Carlo simulations and a real-world dataset. The results show that the point-line cooperative visual-inertial odometry (PL-CVIO) outperforms the independent MSCKF and our previous work CVIO in both low-feature and rich-feature environments.**

I. INTRODUCTION AND RELATED WORK

Simultaneous localization and mapping (SLAM) has received considerable attention in the past few decades and has already been the core technology in many robotics and computer vision applications, such as augmented/virtual reality, autonomous driving, and robot navigation. In GPS-denied environments, visual-inertial navigation systems (VINS) and related algorithms [1]–[3] have received considerable popularity through utilizing low-cost and lightweight onboard cameras and inertial measurement units (IMUs). However, multiple robots have the ability to accomplish tasks more efficiently and achieve higher accuracy than a single robot [4]. Therefore, a key question for a multi-robot group is how to best utilize the environment information and other robots' information.

In human-made scenarios, lines can be considered good complements to points, especially in low-feature environments where only a few point features can be extracted. There are two main categories of methods for processing points and lines in VINS: *indirect* (feature-based) and *direct* methods. In particular, the indirect methods pre-process image flows by extracting feature descriptors and matching them along a sequence [1], [5]–[7]. The indirect methods optimize the system by minimizing the geometric error. The direct methods skip the feature extraction step and optimize the photometric error using row pixels directly [8]–[10]. The direct methods are highly efficient but need to assume brightness constancy (ignoring exposure changes), while the exposure varies heavily in the real-world environment.



(a)

(b)

Fig. 1. **(a)** Overview of the PL-CVIO. Here multiple robots observe point (square) and line (line segment) features in the same environment, neighbors communicate and share common points (green and orange squares) and common lines (orange line segment), and PL-CVIO is performed to estimate the global poses of each robot. **(b)** Point and line feature detection of three different robots in the TUM dataset [11]. Here a green edge denotes a line extracted from the current frame, and a blue dot surrounded by a red square denotes a point extracted from the current frame.

Among the previous feature-based VINS literature, the solutions can be broadly classified into two categories: filter-based methods [7], [12]–[17] and graph-based methods [1], [5], [6], [18]–[20]. One of the *state-of-the-art* works of the filter-based methods is the multi-state constraint Kalman filter (MSCKF) [7], which formed a multi-constraint update by using the measurements of the same feature. A tightly coupled monocular graph-based VIO (VINS-Mono) and non-linear optimization with robust initialization introduced in [1]. Besides, there are also some VINS algorithms using both point and line features. The point-line visual-inertial odometry (PL-VIO) [6] is an extension of VINS-Mono, which can optimize the re-projection errors of the point and line features in a sliding window. PL-SLAM [19] proposed a point-line SLAM framework based on ORB-SLAM [20]. Line features used in Plücker representation for rolling-shutter cameras were designed in [15]. Article [16] proposed two line triangulation algorithms. The analysis of three different line representations (Plücker, Quaternion, Closest

This work was supported by National Science Foundation under Grant CMMI-2027139.

Y. Zhang, P. Zhu, and W. Ren are with the Department of Electrical and Computer Engineering, University of California, Riverside, CA, 92521, USA. Email: {yzhan831, pzhu008}@ucr.edu, ren@ee.ucr.edu

Point) and the corresponding observabilities were provided in [17]. However, all of the above references focus on the single robot case.

One advantage of the cooperative VINS (C-VINS) is the sharing of common features from multiple robots so as to introduce more geometric constraints of the common features. In particular, each robot in the group not only observes its own measurements like in the previous literature, but also collects measurements from the multi-robot group. The robot applies an update to improve the localization performance by utilizing the common feature constraints. There exist some centralized multi-robot solutions [21]–[23]. They usually require expensive computation and communication. Distributed algorithms offer some benefits in this regard. Recently, [24] provided a distributed point-line cooperative SLAM (C-SLAM) algorithm by adopting the M-Space representation of different kinds of features, but the consistency of the estimation cannot be guaranteed because of repeated usage of the same information in the robot group. In [25], each robot in the group processed its own available measurements, and fused the estimation and covariance with other robots within the communication range only at a particular time step. DOOR-SLAM [26] introduced a fully distributed C-SLAM algorithm that contains a pose graph optimizer model and a data-efficient distributed SLAM frontend similar to [27]. Article [28] proposed a fully distributed algorithm using the maximum a posteriori (MAP). Our previous work CVIO [29] provided a fully distributed cooperative algorithm and can guarantee consistency by utilizing the covariance intersection (CI) update, but the low-feature environments were not considered.

In this paper, we propose a fully distributed multi-robot pose estimation algorithm using both point and line features. Each robot not only exploits its own point and line measurements, but also resorts to the cooperation with neighbors (see Fig. 1). Especially in low-feature environments, where robust landmarks are absent, each robot's pose can be estimated with high accuracy by fusing independent point and line features from itself and utilizing the CI update to exploit the constraints imposed by commonly observed point and line features by neighbors. The PL-CVIO algorithm is developed in the *state-of-the-art* OpenVINS [30] system using the monocular camera-IMU architecture. Monte-Carlo simulations and real-world experiments are used to validate the performance of our PL-CVIO algorithm. In both low-feature and rich-feature environments, our algorithm is shown to achieve more accurate localization.

II. PROBLEM FORMULATION

The goal of the cooperative point and line visual-inertial estimator is to track the 3D pose of each robot $\{I_i\}$, for $i = 1, \cdots, n$ in the global frame $\{G\}$. Unlike the independent case, multiple robots can share common features with neighbors. In this paper, we utilize both common point and common line features to improve the localization accuracy.

A. Visual-Inertial Odometry State Vector

In order to perform the PL-CVIO, the state vector of each robot i is defined as:

$$\mathbf{x}_i = \begin{bmatrix} \mathbf{x}_{I_i}^\top & \mathbf{x}_{Calib_i}^\top & \mathbf{x}_{C_i}^\top & t_{d_i} \end{bmatrix}^\top, \tag{1}$$

where $\mathbf{x}_{I_i}$ denotes the IMU state vector, $\mathbf{x}_{Calib_i}$ denotes the rigid body tranformation between the IMU frame and camera frame, $\mathbf{x}_{C_i}$ represents the cloned IMU states, and $t_{d_i} = t_{C_i} - t_{I_i}$ denotes the time-offset between robot i's camera $\{C_i\}$ clock and IMU clock, which treats the IMU clock as the true time [31], [32]. At any time step k, the state vector of each IMU can be writen as:

$$\mathbf{x}_{I_{i,k}} = \begin{bmatrix} {}^{I_{i,k}}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_{i,k}}^\top & {}^{G}\mathbf{v}_{I_{i,k}}^\top & \mathbf{b}_{g_{i,k}}^\top & \mathbf{b}_{a_{i,k}}^\top \end{bmatrix}^\top, \tag{2}$$

where ${}^{I_{i,k}}_{G}\bar{q}$ denotes the JPL unit quaternion [33] representing the rotation from the global frame to the IMU frame at time step k. ${}^{G}\mathbf{p}_{I_{i,k}}$ and ${}^{G}\mathbf{v}_{I_{i,k}}$ are the IMU position and velocity in the global frame at time step k. $\mathbf{b}_{g_{i,k}}$ and $\mathbf{b}_{a_{i,k}}$ are the gyroscope and accelerometer biases at time step k. Then, the error state of the IMU is defined as:

$$\tilde{\mathbf{x}}_{I_{i,k}} = \begin{bmatrix} \delta^{I_{i,k}}_{G}\boldsymbol{\theta}^\top & {}^{G}\tilde{\mathbf{p}}_{I_{i,k}}^\top & {}^{G}\tilde{\mathbf{v}}_{I_{i,k}}^\top & \tilde{\mathbf{b}}_{g_{i,k}}^\top & \tilde{\mathbf{b}}_{a_{i,k}}^\top \end{bmatrix}^\top, \tag{3}$$

where the position, velocity, and bias errors utilize the standard additive error, while the quaternion error state is described by

$$\bar{q} = \delta\bar{q} \otimes \hat{\bar{q}} \simeq \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta}^\top & 1 \end{bmatrix}^\top \otimes \hat{\bar{q}}, \tag{4}$$

where $\hat{(\cdot)}$ denotes the estimate, and $\otimes$ is the quaternion multiplication operator.

In addition to robot i's IMU state, the spatial calibration between its IMU frame and camera frame will also be estimated. In particular, the calibration state vector contains the unit quaternion rotation from the IMU frame to the camera frame ${}^{C_i}_{I_i}\bar{q}$, and the translation from the IMU frame to the camera frame ${}^{C_i}\mathbf{p}_{I_i}$ as:

$$\mathbf{x}_{Calib_i} = \begin{bmatrix} {}^{C_i}_{I_i}\bar{q}^\top & {}^{C_i}\mathbf{p}_{I_i}^\top \end{bmatrix}^\top. \tag{5}$$

Robot i maintains a sliding window with m cloned IMU poses at time step k written as:

$$\mathbf{x}_{C_{i,k}} = \begin{bmatrix} {}^{I_{i,k-1}}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_{i,k-1}}^\top & \cdots & {}^{I_{i,k-m}}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_{i,k-m}}^\top \end{bmatrix}^\top. \tag{6}$$

B. Dynamic System Model

For each robot i, the measurement of the IMU linear acceleration ${}^{I_i}\mathbf{a}_m$ and the angular velocity ${}^{I_i}\boldsymbol{\omega}_m$ are modeled as:

$${}^{I_i}\mathbf{a}_m = {}^{I_i}\mathbf{a} + {}^{I_i}_{G}\mathbf{R}\,{}^{G}\mathbf{g} + \mathbf{b}_{a_i} + \mathbf{n}_{a_i}, \tag{7}$$

$${}^{I_i}\boldsymbol{\omega}_m = {}^{I_i}\boldsymbol{\omega} + \mathbf{b}_{g_i} + \mathbf{n}_{g_i}, \tag{8}$$

where ${}^{I_i}\mathbf{a}$ and ${}^{I_i}\boldsymbol{\omega}$ are the true angular velocity and linear acceleration. $\mathbf{n}_{a_i}$ and $\mathbf{n}_{g_i}$ represent the continuous-time Gaussian noises that contaminate the IMU measurements.

${}^{G}\mathbf{g}$ denotes the gravity expressed in the global frame. Then, the dynamic system of each IMU can be modeled as [33]:

$$\begin{aligned} {}^{I_i}_{G}\dot{\bar{q}}(t) &= \frac{1}{2}\mathbf{\Omega}\left({}^{I_i}\boldsymbol{\omega}(t)\right){}^{I_i}_{G}\bar{q}(t), \quad \dot{\mathbf{b}}_{g_i}(t) = \mathbf{n}_{wg_i}(t), \\ {}^{G}\dot{\mathbf{v}}_{I_i}(t) &= {}^{G}\mathbf{a}_i(t), \quad \dot{\mathbf{b}}_{a_i}(t) = \mathbf{n}_{wa_i}(t), \quad {}^{G}\dot{\mathbf{p}}_{I_i}(t) = {}^{G}\mathbf{v}_{I_i}(t) \end{aligned} \tag{9}$$

where ${}^{G}\mathbf{a}_i$ is the body acceleration in the global frame. ${}^{G}\mathbf{v}_{I_i}$, ${}^{G}\mathbf{p}_{I_i}$ are the velocity and position of the IMU in the global frame. $\mathbf{n}_{wg_i}$ and $\mathbf{n}_{wa_i}$ denote the zero-mean Gaussian noises driving the IMU biases. $\boldsymbol{\omega} = [\omega_x \ \omega_y \ \omega_z]^\top$ is the rotational velocity in the IMU frame and

$$\mathbf{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor\boldsymbol{\omega}\times\rfloor & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix}, \lfloor\boldsymbol{\omega}\times\rfloor = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}.$$

After linearization, the continuous-time IMU error-state can be written as:

$$\dot{\tilde{\mathbf{x}}}_i(t) \simeq \mathbf{F}_i(t)\tilde{\mathbf{x}}_i(t) + \mathbf{G}_i(t)\mathbf{n}_i(t), \tag{10}$$

where $\mathbf{F}_i(t)$ is the 15×15 continuous-time IMU error-state Jacobian matrix, $\mathbf{G}_i(t)$ is the 15×12 noise Jacobian matrix, and $\mathbf{n}_i(t) = \begin{bmatrix}\mathbf{n}_{g_i}^\top & \mathbf{n}_{wg_i}^\top & \mathbf{n}_{a_i}^\top & \mathbf{n}_{wa_i}^\top\end{bmatrix}^\top$ is the system noise with the covariance matrix $\mathbf{Q}_i$.

In order to propagate the covariance matrix from discrete-time t_k to t_{k+1}, the state transition matrix $\mathbf{\Phi}_i\left(t_{k+1}, t_k\right)$ is computed by solving the differential equation:

$$\dot{\mathbf{\Phi}}_i\left(t_{k+1}, t_k\right) = \mathbf{F}_i\mathbf{\Phi}_i\left(t_{k+1}, t_k\right), \tag{11}$$

with the initial condition $\mathbf{\Phi}_i\left(t_k, t_k\right) = \mathbf{I}_{15}$. Thus, the discrete-time noise covariance can be expressed as:

$$\mathbf{Q}_{i,k} = \int_{t_k}^{t_{k+1}} \mathbf{\Phi}_i(t_{k+1},\tau)\mathbf{G}_i(\tau)\mathbf{Q}_i\mathbf{G}_i^\top(\tau)\mathbf{\Phi}_i(t_{k+1},\tau)^\top \mathbf{d}\tau, \tag{12}$$

and the propagated covariance can be written as:

$$\mathbf{P}_{i,k+1|k} = \mathbf{\Phi}_i\left(t_{k+1}, t_k\right)\mathbf{P}_{i,k|k}\mathbf{\Phi}_i\left(t_{k+1}, t_k\right)^\top + \mathbf{Q}_{i,k}. \tag{13}$$

C. Point and Line Measurement Models

In low-feature environments, lines are good complements to points. Hence we consider both point and line measurements in this paper. The point measurements of robot i can be described by:

$${}^{C_i}\mathbf{z}_p = \Pi\left({}^{C_i}\mathbf{x}_p\right) + \mathbf{w}_{p_i}, \quad \Pi\left(\begin{bmatrix}x & y & z\end{bmatrix}^\top\right) = \begin{bmatrix}\frac{x}{z} & \frac{y}{z}\end{bmatrix}^\top, \tag{14}$$

where ${}^{C_i}\mathbf{x}_p$ is the 3D position of the point in the camera frame, and $\mathbf{w}_{p_i}$ denotes the corresponding measurement noise. Based on the relative transformation and time offset definition in (1), the relationship between point feature in the global frame ${}^{G}\mathbf{x}_p$ and in the camera frame ${}^{C_i}\mathbf{x}_p$ can be expressed as:

$${}^{C_i}\mathbf{x}_p = {}^{C_i}_{I_i}\mathbf{R}{}^{I_i}_{G}\mathbf{R}\left(\bar{t}_i\right)\left({}^{G}\mathbf{x}_p - {}^{G}\mathbf{p}_{I_i}\left(\bar{t}_i\right)\right) + {}^{C_i}\mathbf{p}_{I_i}, \tag{15}$$

where $\bar{t}_i = t_i - t_{d_i}$ is the exact camera time of the relative transformation between the global frame and the IMU frame.

For a 3D line, we adopt the *Closest Point* representation [16], which represents the 3D line by multiplying a unit quaternion and the corresponding distance scalar from the origin to this line. Given the 3D positions of two points $\mathbf{p_{f1}}$ and $\mathbf{p_{f2}}$ on a line, the Plücker coordinate can be expressed by [34]:

$$\begin{bmatrix}\mathbf{n}_l \\ \mathbf{v}_l\end{bmatrix} = \begin{bmatrix}\lfloor\mathbf{p_{f1}}\times\rfloor\,\mathbf{p_{f2}} \\ \mathbf{p_{f2}} - \mathbf{p_{f1}}\end{bmatrix}, \tag{16}$$

where $\mathbf{n}_l$ denotes the normal direction of the line-plane and $\mathbf{v}_l$ is the line direction. Then, the *Closest Point* line can be expressed as:

$${}^{G}\mathbf{x}_l = d_l\bar{q}_l = \begin{bmatrix}\mathbf{q}_l^\top & q_l\end{bmatrix}^\top, \tag{17}$$

where the distance scalar can be computed as $d_l = \|\mathbf{n}_l\| / \|\mathbf{v}_l\|$. The unit quaternion $\bar{q}_l$ can be transformed from $\mathbf{R}\left(\bar{q}_l\right) = [\mathbf{n_e}\ \mathbf{v_e}\ \lfloor\mathbf{n_e}\times\rfloor\mathbf{v_e}]$, where $\mathbf{n_e}$ and $\mathbf{v_e}$ are the unit 3D vectors of $\mathbf{n}_l$ and $\mathbf{v}_l$.

Moreover, for each robot i, we adopt the simple projective line measurement model [35] to describe the 2D line distance from two line endpoints, $\mathbf{x}_{s_i} = [u_{s_i}\ v_{s_i}\ 1]^\top$ and $\mathbf{x}_{e_i} = [u_{e_i}\ v_{e_i}\ 1]^\top$ to the 2D line segment:

$${}^{C_i}\mathbf{z}_l = \begin{bmatrix}\frac{\mathbf{x}_{s_i}^\top\mathbf{l}_i}{\sqrt{l_1^2+l_2^2}} & \frac{\mathbf{x}_{e_i}^\top\mathbf{l}_i}{\sqrt{l_1^2+l_2^2}}\end{bmatrix}^\top, \tag{18}$$

where $\mathbf{l}_i = [l_1\ l_2\ l_3]^\top$ denotes the 2D line representation. The line measurement can be projected from the 3D line in the camera frame as in [17]:

$$\begin{bmatrix}l_1 \\ l_2 \\ l_3\end{bmatrix} = \begin{bmatrix}f_{v_i} & 0 & 0 & 0 & 0 & 0 \\ 0 & f_{u_i} & 0 & 0 & 0 & 0 \\ -f_{v_i}c_{u_i} & -f_{u_i}c_{v_i} & f_{u_i}f_{v_i} & 0 & 0 & 0\end{bmatrix}{}^{C_i}\mathbf{L}, \tag{19}$$

where f_{u_i}, f_{v_i}, c_{u_i}, c_{v_i} are the camera intrinsic parameters, and ${}^{C_i}\mathbf{L} = \begin{bmatrix}{}^{C_i}d_l{}^{C_i}\mathbf{n}_e^\top & {}^{C_i}\mathbf{v}_e^\top\end{bmatrix}^\top$ is the Plücker coordinate representation of the 3D line in the camera frame. The line transformation from the global frame to the camera frame can be written as:

$${}^{C_i}\mathbf{L} = \begin{bmatrix}{}^{C_i}_{I_i}\mathbf{R} & \lfloor{}^{C_i}\mathbf{P}_{I_i}\times\rfloor{}^{C_i}_{I_i}\mathbf{R} \\ \mathbf{0}_3 & {}^{C_i}_{I_i}\mathbf{R}\end{bmatrix}{}^{I_i}\mathbf{L}$$

and

$${}^{I_i}\mathbf{L} = \begin{bmatrix}{}^{I_i}_{G}\mathbf{R}\left(\bar{t}_i\right) & -{}^{I_i}_{G}\mathbf{R}\left(\bar{t}_i\right)\lfloor{}^{G}\mathbf{P}_{I_i}\left(\bar{t}_i\right)\times\rfloor \\ \mathbf{0}_3 & {}^{I_i}_{G}\mathbf{R}\left(\bar{t}_i\right)\end{bmatrix}{}^{G}\mathbf{L}, \tag{20}$$

where ${}^{I_i}\mathbf{L}$ and ${}^{G}\mathbf{L}$ are the Plücker line representations in the IMU frame and the global frame, respectively.

D. Independent Point and Line Feature Update

To perform the independent point or line feature update, a standard MSCKF update [7] will be applied to each robot. In particular, we collect all of the point and line measurements over the current sliding window. By stacking the measurements of one point or line, we can triangulate the point feature or line feature utilizing the estimate of the IMU poses. To simplify the notation, let $\bar{\mathbf{x}}_f$ denotes either a

point feature or a line feature, and the measurement residual of robot i can be linearized as:

$$\mathbf{r}_i = \mathbf{h}\left(\tilde{\mathbf{x}}_i, {}^G\tilde{\mathbf{x}}_f\right) + \mathbf{w}_i \simeq \mathbf{H}_{i,x}\tilde{\mathbf{x}}_i + \mathbf{H}_{i,f}{}^G\tilde{\mathbf{x}}_f + \mathbf{w}_i, \quad (21)$$

where $\mathbf{r}_i$ is the residual of a point or line measurement. $\mathbf{H}_{i,x}$ and $\mathbf{H}_{i,f}$ denote the Jacobians w.r.t. the state vector and the feature, respectively. $\mathbf{w}_i$ denotes the noise vector corresponding to the point or line feature.

After that, we perform the left nullspace projection by applying the QR decompositon to $\mathbf{H}_{i,f}$ in (21) as:

$$\begin{bmatrix} \mathbf{r}_i^1 \\ \mathbf{r}_i^2 \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{i,x}^1 \\ \mathbf{H}_{i,x}^2 \end{bmatrix} \tilde{\mathbf{x}}_i + \begin{bmatrix} \mathbf{H}_{i,f}^1 \\ \mathbf{0} \end{bmatrix} {}^G\tilde{\mathbf{x}}_f + \begin{bmatrix} \mathbf{w}_i^1 \\ \mathbf{w}_i^2 \end{bmatrix}. \quad (22)$$

In this expression, $\mathbf{r}_i^2$ is only related to the state vector $\tilde{\mathbf{x}}_i$. Hence robot i will perform an EKF update using $\mathbf{r}_i^2$, while $\mathbf{r}_i^1$ will be dropped.

E. Common Point and Line Feature Update

Note that neighboring robots might observe a common point or line feature. Hence, we will further exploit both point and line feature constraints among neighbors to improve the localization accuracy. The robots can communicate with their neighbors to share information.

Robot i and its neighbors will apply the linearization (21) and the left nullspace projection (22) to the common feature, denoted as ${}^G\tilde{\mathbf{x}}_f$. As in Sec II-D, robot i will use $\mathbf{r}_i^2$ for an EKF update. However, instead of dropping $\mathbf{r}_i^1$, robot i will exploit shared information from its neighbors. It will construct a new residual system that depends on the common point or line feature ${}^G\tilde{\mathbf{x}}_f$ by stacking the top parts in (22) associated with itself and its neighbors as in [29]:

$$\begin{bmatrix} \mathbf{r}_i^1 \\ \mathbf{r}_{i_1}^1 \\ \vdots \\ \mathbf{r}_{i_j}^1 \end{bmatrix} = diag\left(\begin{bmatrix} \mathbf{H}_{i,x}^1 \\ \mathbf{H}_{i_1,x}^1 \\ \vdots \\ \mathbf{H}_{i_j,x}^1 \end{bmatrix}\right) \begin{bmatrix} \tilde{\mathbf{x}}_i \\ \tilde{\mathbf{x}}_{i_1} \\ \vdots \\ \tilde{\mathbf{x}}_{i_j} \end{bmatrix} + \begin{bmatrix} \mathbf{H}_{i,f}^1 \\ \mathbf{H}_{i_1,f}^1 \\ \vdots \\ \mathbf{H}_{i_j,f}^1 \end{bmatrix} {}^G\tilde{\mathbf{x}}_f + \begin{bmatrix} \mathbf{w}_i^1 \\ \mathbf{w}_{i_1}^1 \\ \vdots \\ \mathbf{w}_{i_j}^1 \end{bmatrix}, \quad (23)$$

where $diag$ denotes the block-diagonal matrix, and $i_1 \dots i_j$ denote the neighbors of robot i. Then, we utilize the left nullspace projection to the stacked common point or line feature Jacobian matrix in (23) and obtain a new residual system that is independent of the common feature as:

$$\mathbf{r}_i' = \begin{bmatrix} \mathbf{H}_{i,x}' & \mathbf{H}_{i_1,x}' & \cdots & \mathbf{H}_{i_j,x}' \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_i \\ \tilde{\mathbf{x}}_{i_1} \\ \vdots \\ \tilde{\mathbf{x}}_{i_j} \end{bmatrix} + \mathbf{w}_i'. \quad (24)$$

In order to guarantee the consistency of estimation, we adopt the CI-EKF algorithm in [29], where the weights of the CI are $\omega_i > 0$, $\omega_{i_l} > 0$, and $\omega_i + \sum_{l=1}^{j} \omega_{i_l} = 1$. The Kalman gain of robot i is given by:

$$\mathbf{K}_i = \frac{\mathbf{P}_{i,k+1|k}\mathbf{H}_{i,x}'^{\top}}{\omega_i} \left(\sum_{r \in \mathcal{N}_i} \frac{1}{\omega_r} \mathbf{H}_{r,x}' \mathbf{P}_{r,k+1|k} \mathbf{H}_{r,x}'^{\top} + \mathbf{R}_i \right)^{-1}, \quad (25)$$

where $\mathcal{N}_i$ denotes the set of robot i's neighboring robots that the current common feature can be tracked, and $\mathbf{R}_i$ denote the covariance matrix associated with $\mathbf{w}_i'$. Then, the state correction of robot i can be written as:

$$\Delta\mathbf{x}_{i,k} = \mathbf{K}_i \mathbf{r}_i'. \quad (26)$$

The state covariance matrix of robot i is updated using the CI as:

$$\mathbf{P}_{i,k+1|k+1} = \frac{1}{\omega_i} \left(\mathbf{I} - \mathbf{K}_i \mathbf{H}_{i,x}'\right) \mathbf{P}_{i,k+1|k}. \quad (27)$$

III. SIMULATIONS AND EXPERIMENTS

In this section, we utilize Monte-Carlo simulations and real-world datasets to verify that common line features can improve localization accuracy in cooperative cases, and line features can also improve the accuracy in independent cases. We compare our PL-CVIO algorithm with the previous works in Table I under two different environments, where *low-feature* scenes contain a few features and *rich-feature* scenes contain enough features. As shown in Table. I, P-VIO denotes the independent MSCKF algorithm [7], PL-VIO denotes the independent point-line MSCKF algorithm [16], P-CVIO denotes our previous work CVIO [29], IPL-CP-CVIO denotes the algorithm which not only utilizes independent point-line features from each robot but also collects the common point features from the neighbors, and PL-CVIO uses both independent and common point-line features as in this paper.

TABLE I
Descriptions of various algorithms to be compared in the simulations and experiments.

Algorithm	Independent Features	Common Features
P-VIO [7]	Points	✗
PL-VIO [16]	Points w/ Lines	✗
P-CVIO [29]	Points	Points
IPL-CP-CVIO	Points w/ Lines	Points
PL-CVIO	Points w/ Lines	Points w/ Lines

A. Monte-Carlo Simulations

For our Monte-Carlo simulations, we utilize a group of three robots. *Robot 0* in the group follows the real trajectory of a dataset, and the trajectories of *robot 1* and *robot 2* are created by adding position and orientation offsets to the real one. After that, the 3D features and the corresponding 2D measurements are generated if the number of the point or line

TABLE II

The RMSE of the orientation / position (degrees / meters) of three robots using three different algorithms in different EuRoC datasets. **R** denotes the rich-feature environments with enough point-line features, and **L** denotes the low-feature cases. The average denotes mean of all three rooms per algorithm per robot per environment (rich-feature/low-feature). R0, R1, and R2 represent three robots following three different trajectories in each environment.

	V1_01_R	**V1_02_R**	**V1_03_R**	**Average**	**V1_01_L**	**V1_02_L**	**V1_03_L**	**Average**
R0 P-VIO	0.481 / 0.260	0.621 / 0.064	0.874 / 0.061	0.659 / 0.128	1.277 / 0.483	0.717 / 0.177	1.184 / 0.684	1.060 / 0.448
R0 P-CVIO	0.091 / 0.056	0.157 / 0.022	0.118 / 0.027	0.122 / 0.035	0.524 / 0.148	0.449 / 0.080	0.743 / 0.299	0.572 / 0.176
R0 PL-CVIO	0.090 / 0.047	0.147 / 0.021	0.101 / 0.025	0.113 / 0.031	0.159 / 0.078	0.167 / 0.064	0.182 / 0.099	0.169 / 0.080
R1 P-VIO	1.166 / 0.205	0.167 / 0.049	0.419 / 0.049	0.584 / 0.101	0.888 / 0.152	0.785 / 0.170	0.775 / 0.150	0.816 / 0.157
R1 P-CVIO	0.104 / 0.060	0.183 / 0.026	0.127 / 0.026	0.138 / 0.037	0.584 / 0.137	0.613 / 0.130	0.733 / 0.075	0.643 / 0.144
R1 PL-CVIO	0.089 / 0.052	0.176 / 0.021	0.096 / 0.026	0.120 / 0.033	0.213 / 0.092	0.231 / 0.078	0.249 / 0.074	0.231 / 0.081
R2 P-VIO	0.960 / 0.132	0.230 / 0.078	0.325 / 0.062	0.505 / 0.091	1.589 / 0.596	1.493 / 0.195	0.676 / 0.202	1.253 / 0.331
R2 P-CVIO	0.099 / 0.056	0.170 / 0.023	0.123 / 0.025	0.131 / 0.035	0.603 / 0.161	0.690 / 0.179	0.538 / 0.165	0.610 / 0.168
R2 PL-CVIO	0.095 / 0.056	0.167 / 0.022	0.109 / 0.021	0.127 / 0.033	0.150 / 0.096	0.167 / 0.081	0.182 / 0.073	0.166 / 0.083

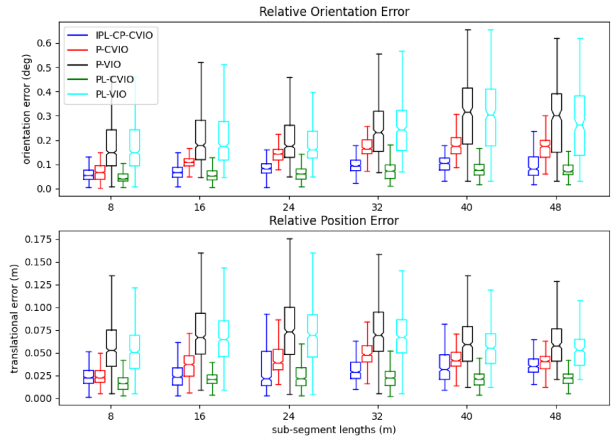


Fig. 2. Boxplot of the statistics of the Monte-Carlo simulation under the rich-feature Udel_gore environment by extracing 150 points per frame, and 50 lines if the line update is used.

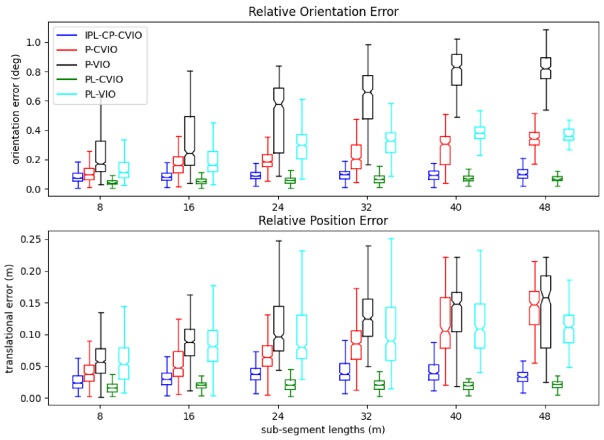


Fig. 3. Boxplot of the statistics of the Monte-Carlo simulation under low-feature Udel_gore environment by extracing 50 points per frame, and 50 lines if the line update is used.

measurements is below the threshold in the current frame. Then, the constraints of the same feature from one robot and the constraints of the common features from neighbors are collected and utilized to update the current state.

The low-feature and rich-feature environments are divided by extracting different numbers of point features. In the rich-feature environments, the number of point features is 150 and the number of line features is 50 in each frame. We reduce the number of point features to 50 for the low-feature cases. For both of these two environments, we utilize the First-Estimation Jacobian (FEJ) and online camera-IMU calibration [30]. After running 30 Monte-Carlo loops, the statistics of the relative orientation error (ROE) and the relative position error (RPE) under the rich-feature or low-feature Udel_gore dataset are shown in Fig. 2 and Fig. 3, respectively. We can see that our PL-CVIO algorithm outperforms all other algorithms in both environments. Especially in the low-feature case, we can find out that the common line can reduce the ROE and RPE obviously (blue and red bar) as in Fig. 3. Moreover, an interesting discovery is that PL-VIO outperforms P-CVIO if a limited number of points are observed in each frame. In this case, the number of common point features is also limited, and hence the cooperative method P-CVIO that relies on only common point features has limited resources to resort to. In contrast, the methods PL-VIO and PL-CVIO that further exploit line features exhibit better performance while PL-CVIO achieves the best performance as it exploits not only point and line features but also cooperation with neighbors.

Additionally, we simulate our PL-CVIO algorithm in all of the EuRoC V1 datasets [36] and compare it with P-VIO and P-CVIO in both low-feature and rich-feature environments.

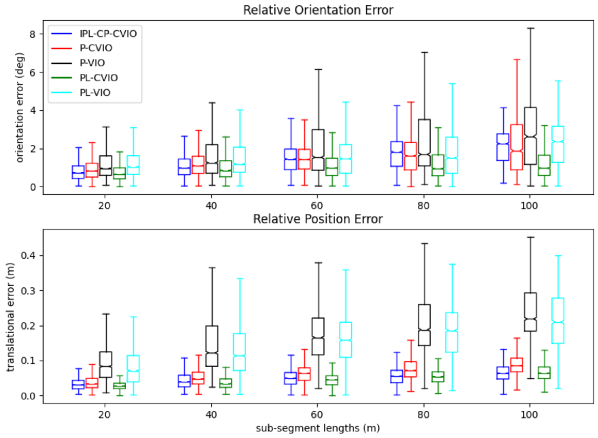


Fig. 4. Boxplot of the result of *Robot 0* (Room 1) in the TUM Visual-Inertial Dataset by extracing 200 points per frame, and 50 lines if the line update is used.

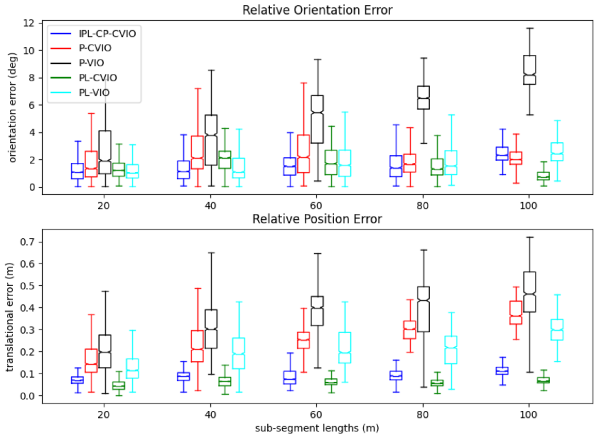


Fig. 5. Boxplot of the result of *Robot 0* (Room 1) in the TUM Visual-Inertial Dataset by extracing 50 points per frame, and 50 lines if the line update is used.

The RMSE of the orientation and position of each robot and the mean RMSE of each algorithm in each environment are recorded in Table II. The RMSE results show that our PL-CVIO algorithm outperforms P-CVIO and P-VIO in all simulated scenarios. Especially in low-feature environments, the PL-CVIO improves the RMSE of orientation and position dramatically.

B. Experiments

For the real-world experiments, the position and orientation of each robot are initialized corresponding to the ground truth. The point features are extracted from each frame using FAST [37], and are tracked crossing frames or matched with the point observations from other robot utilizing ORB [38] with an 8-point RANSAC algorithm. At the same time, line segments are extracted by leveraging the LSD [39] and tracked by LBD [40]. Besides, we add some outlier elimination strategies to remove the line segment where (1) the LBD distance is larger than 50; (2) the length of the line segment is smaller than 50 pixels; (3) the distance between the origin and this line is smaller than 0.1 or larger than 100; (4) the line disparity is too small to avoid singularity when applying the SVD [41] to triangulate this line.

We evaluate our PL-CVIO algorithm in the TUM Visual-Inertial Dataset Rooms 1, 3, and 5 [11], where the IMU frequency is 200 Hz and the camera frequency is 20Hz. We load all three datasets of the same room and run all five algorithms with three robots separately. Besides, we extract a different number of point features to imitate low-feature and rich-feature environments. As a result, we show the experimental results of our PL-CVIO algorithm compared with the other four algorithms in respectively rich-feature environments as in Fig. 4 and low-feature environments as in Fig. 5. We also show the RMSE of the orientation and position of each robot by utilizing different algorithms in the TUM dataset as in Table III. From the ROE/RPE and

TABLE III

The RMSE of the orientation / position (degrees / meters) of three robots under the low-feature environments by using five different algorithms in the TUM Visual-Inertial dataset.

Algorithm	Robot 0	Robot 1	Robot 2
P-VIO	7.473 / 0.442	4.357 / 0.313	5.468 / 0.372
PL-VIO	2.367 / 0.295	1.798 / 0.254	1.872 / 0.240
P-CVIO	2.301 / 0.377	3.824 / 0.267	2.616 / 0.263
IPL-CP-CVIO	1.905 / 0.098	1.722 / 0.115	1.542 / 0.095
PL-CVIO	1.349 / 0.061	1.665 / 0.086	1.379 / 0.067

the RMSE results, it is clear that line features can improve the accuracy of P-VIO and the common point-line features can improve the performance of the P-CVIO. Besides, the lines improve the performance obviously in the low-feature scenes by comparing the P-VIO and PL-VIO, as well as P-CVIO and IPL-CP-CVIO in Table. III. Additionally, our PL-CVIO outperforms all four algorithms in all experiment cases.

IV. CONCLUSIONS

In this paper, we have proposed a fully distributed point-line cooperative visual-inertial navigation system. We compared the performance of the proposed algorithm with four other algorithms under rich-feature or low-feature environments in both Monte-Carlo simulations and real-world datasets. All of the results indicated that our PL-CVIO outperformed the independent MSCKF and CVIO. Also, we verified that the line feature can improve the accuracy of localization in independent cases, and the common line features can perform better in cooperative cases.

REFERENCES

[1] Qin, T., Li, P. and Shen, S., 2018. Vins-mono: A robust and versatile monocular visual-inertial state estimator. IEEE Transactions on Robotics, 34(4), pp.1004-1020.

[2] Li, M. and Mourikis, A.I., 2013. High-precision, consistent EKF-based visual-inertial odometry. The International Journal of Robotics Research, 32(6), pp.690-711.

[3] Wu, K., Ahmed, A.M., Georgiou, G.A. and Roumeliotis, S.I., 2015, July. A Square Root Inverse Filter for Efficient Vision-aided Inertial Navigation on Mobile Devices. In Robotics: Science and Systems (Vol. 2).

[4] Huang, G.P., Trawny, N., Mourikis, A.I. and Roumeliotis, S.I., 2009, January. On the consistency of multi-robot cooperative localization. In Robotics: Science and Systems (pp. 65-72).

[5] Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R. and Furgale, P., 2015. Keyframe-based visual–inertial odometry using nonlinear optimization. The International Journal of Robotics Research, 34(3), pp.314-334.

[6] He, Y., Zhao, J., Guo, Y., He, W. and Yuan, K., 2018. Pl-vio: Tightly-coupled monocular visual–inertial odometry using point and line features. Sensors, 18(4), p.1159.

[7] Mourikis, A.I. and Roumeliotis, S.I., 2007, April. A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation. In ICRA (Vol. 2, p. 6).

[8] Engel, J., Schöps, T. and Cremers, D., 2014, September. LSD-SLAM: Large-scale direct monocular SLAM. In European conference on computer vision (pp. 834-849). Springer, Cham.

[9] Zhou, L., Wang, S. and Kaess, M., 2021. DPLVO: Direct Point-Line Monocular Visual Odometry. IEEE Robotics and Automation Letters, 6(4), pp.7113-7120.

[10] Engel, J., Koltun, V. and Cremers, D., 2017. Direct sparse odometry. IEEE transactions on pattern analysis and machine intelligence, 40(3), pp.611-625.

[11] Sturm, J., Engelhard, N., Endres, F., Burgard, W. and Cremers, D., 2012, October. A benchmark for the evaluation of RGB-D SLAM systems. In 2012 IEEE/RSJ international conference on intelligent robots and systems (pp. 573-580). IEEE.

[12] Bloesch, M., Omari, S., Hutter, M. and Siegwart, R., 2015, September. Robust visual inertial odometry using a direct EKF-based approach. In 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS) (pp. 298-304). IEEE.

[13] Forster, C., Pizzoli, M. and Scaramuzza, D., 2014, May. SVO: Fast semi-direct monocular visual odometry. In 2014 IEEE international conference on robotics and automation (ICRA) (pp. 15-22). IEEE.

[14] Paul, M.K., Wu, K., Hesch, J.A., Nerurkar, E.D. and Roumeliotis, S.I., 2017, May. A comparative analysis of tightly-coupled monocular, binocular, and stereo VINS. In 2017 IEEE International Conference on Robotics and Automation (ICRA) (pp. 165-172). IEEE.

[15] Yu, H. and Mourikis, A.I., 2015, September. Vision-aided inertial navigation with line features and a rolling-shutter camera. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 892-899). IEEE.

[16] Yang, Y., Geneva, P., Eckenhoff, K. and Huang, G., 2019, November. Visual-inertial odometry with point and line features. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 2447-2454). IEEE.

[17] Yang, Y. and Huang, G., 2019, May. Aided inertial navigation: Unified feature representations and observability analysis. In 2019 International Conference on Robotics and Automation (ICRA) (pp. 3528-3534). IEEE.

[18] Campos, C., Elvira, R., Rodríguez, J.J.G., Montiel, J.M. and Tardós, J.D., 2021. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. IEEE Transactions on Robotics, 37(6), pp.1874-1890.

[19] Pumarola, A., Vakhitov, A., Agudo, A., Sanfeliu, A. and Moreno-Noguer, F., 2017, May. PL-SLAM: Real-time monocular visual SLAM with points and lines. In 2017 IEEE international conference on robotics and automation (ICRA) (pp. 4503-4508). IEEE.

[20] Mur-Artal, R., Montiel, J.M.M. and Tardos, J.D., 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE transactions on robotics, 31(5), pp.1147-1163.

[21] Luna, R. and Bekris, K.E., 2011, September. Efficient and complete centralized multi-robot path planning. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 3268-3275). IEEE.

[22] Liu, C. and Kroll, A., 2012, April. A centralized multi-robot task allocation for industrial plant inspection by using a* and genetic algorithms. In International Conference on Artificial Intelligence and Soft Computing (pp. 466-474). Springer, Berlin, Heidelberg.

[23] Kang, H., Kim, H. and Kwon, Y.M., 2019, December. RECEN: resilient MANET based centralized multi robot system using mobile agent system. In 2019 IEEE symposium series on computational intelligence (SSCI) (pp. 1952-1958). IEEE.

[24] Benedettelli, D., Garulli, A. and Giannitrapani, A., 2012. Cooperative SLAM using M-Space representation of linear features. Robotics and Autonomous Systems, 60(10), pp.1267-1278.

[25] Karam, N., Chausse, F., Aufrere, R. and Chapuis, R., 2006, October. Localization of a group of communicating vehicles by state exchange. In 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 519-524). IEEE.

[26] Lajoie, P.Y., Ramtoula, B., Chang, Y., Carlone, L. and Beltrame, G., 2020. DOOR-SLAM: Distributed, online, and outlier resilient SLAM for robotic teams. IEEE Robotics and Automation Letters, 5(2), pp.1656-1663.

[27] Cieslewski, T., Choudhary, S. and Scaramuzza, D., 2018, May. Data-efficient decentralized visual SLAM. In 2018 IEEE international conference on robotics and automation (ICRA) (pp. 2466-2473). IEEE.

[28] Nerurkar, E.D., Roumeliotis, S.I. and Martinelli, A., 2009, May. Distributed maximum a posteriori estimation for multi-robot cooperative localization. In 2009 IEEE International Conference on Robotics and Automation (pp. 1402-1409). IEEE.

[29] Zhu, P., Yang, Y., Ren, W. and Huang, G., 2021, May. Cooperative visual-inertial odometry. In 2021 IEEE International Conference on Robotics and Automation (ICRA) (pp. 13135-13141). IEEE.

[30] Geneva, P., Eckenhoff, K., Lee, W., Yang, Y. and Huang, G., 2020, May. Openvins: A research platform for visual-inertial estimation. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 4666-4672). IEEE.

[31] Li, M. and Mourikis, A.I., 2014. Online temporal calibration for camera–IMU systems: Theory and algorithms. The International Journal of Robotics Research, 33(7), pp.947-964.

[32] Qin, T. and Shen, S., 2018, October. Online temporal calibration for monocular visual-inertial systems. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 3662-3669). IEEE.

[33] Trawny, N. and Roumeliotis, S.I., 2005. Indirect Kalman filter for 3D attitude estimation. University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep, 2, p.2005.

[34] Zuo, X., Xie, X., Liu, Y. and Huang, G., 2017, September. Robust visual SLAM with point and line features. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 1775-1782). IEEE.

[35] Bartoli, A. and Sturm, P., 2005. Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. Computer vision and image understanding, 100(3), pp.416-441.

[36] Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M.W. and Siegwart, R., 2016. The EuRoC micro aerial vehicle datasets. The International Journal of Robotics Research, 35(10), pp.1157-1163.

[37] Rosten, E., Porter, R. and Drummond, T., 2008. Faster and better: A machine learning approach to corner detection. IEEE transactions on pattern analysis and machine intelligence, 32(1), pp.105-119.

[38] Rublee, E., Rabaud, V., Konolige, K. and Bradski, G., 2011, November. ORB: An efficient alternative to SIFT or SURF. In 2011 International conference on computer vision (pp. 2564-2571). Ieee.

[39] Von Gioi, R.G., Jakubowicz, J., Morel, J.M. and Randall, G., 2008. LSD: A fast line segment detector with a false detection control. IEEE transactions on pattern analysis and machine intelligence, 32(4), pp.722-732.

[40] Zhang, L. and Koch, R., 2013. An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency. Journal of Visual Communication and Image Representation, 24(7), pp.794-805.

[41] Bradski, G., 2000. The openCV library. Dr. Dobb's Journal: Software Tools for the Professional Programmer, 25(11), pp.120-123.