Fusion VIO: An Extendable Sensor Fusion Framework for Unmaned Vehicles

Ruonan Guo, Mengyu Hu, Chang Shao*, Jiayang Zhao
Beijing Sineva Technology Co., Ltd., Beijing 100176, China
DOI: 10.32629/aes.v3i2.894

Abstract: Real-time perception based on the simultaneous localization and mapping (SLAM) technology is of immense application potential in the industry filed. In this work, we provide an extended Kalman filter (EKF) based visual SLAM framework named Fusion VIO, which contains HDR imaging system, hardware synchronization, object detection, filter based sensor fusion, and semantic fusion. Notably, Fusion VIO well fuses loop closure information, which takes effects in solving the error problem during operation of the robot and improving the robustness of the system. Besides, experimental validation shows that Fusion VIO not only maintains high accuracy but also cost one-third resources compared to the common VINS fusion. Our work demonstrates a novel sensor fusion framework for unmaned vehicles.
Keywords: time synchronization, SLAM, sensor fusion

1. Introduction

Simultaneous localization and mapping (SLAM) technology meets the requirements of robots and AR/VR devices to perceive the current scene and position in real-time during operation. In the last decade, new spin-off technologies, such as visual SLAM and visual inertial SLAM, have been proposed. At the same time, chips and MEMS devices for SLAM utilization have made great process as well. In addition, sensors such as cameras, IMUs and lidars have improved at higher precision and lower price also.

The image sensor has received a much attention due to the advantages of tiny, low-cost, and off-the-shelf hardware.. A substantial amount of visual sensor-based SLAM methods and solutions are invented. Visual SLAM solution is perfect for engine efficiency, excellent robustness and provides rich perception information for upper-level applications.

VINS (visual-inertial navigation system) is one of the most promising low-cost synchronous localization and mapping solutions. It has the characteristics of small, light, and low cost. The primary module of VINS includes data acquisition, time synchronization, loop closure, and optimization. In many VINS solutions, the solution equipped with camera and inertial measurement unit (IMU) named visual-inertial SLAM (VI-SLAM). The angular velocity and acceleration measured by the IMU can be complementary to the camera measurements. After the fusion, there is only a slight drift, which can configure a more complete SLAM system. According to back-end optimization methods, VI-SLAM can be characterized into two parts - filter-based VI-SLAM and optimization-based VI-SLAM[1-2].

Open keyframe based visual-inertial SLAM (OKVIS) is an optimization-based method announced by the Swiss Federal Institute of Technology, ASL lab announced [3]; the sliding window based on keyframes perform batch nonlinear optimization, key frames before the sliding window are marginalized out after estimation. Front-end adopt multi scale Harris detector for feature extraction. Furthermore, based on the Harris detector, BRISK[4] (binary robust invariant salable key-point) descriptor is calculated in order to associate data from frame to frame. ORB-SLAM 3 uses three threads that track feature points in real time[5], optimization thread for local mapping (co-visibility graph), and a global optimization thread (essential graph). It can achieve excellent tracking and 3D map relocalization. The map ensures the global consistency between the trajectory of the robot and the map, resulting in better performance.

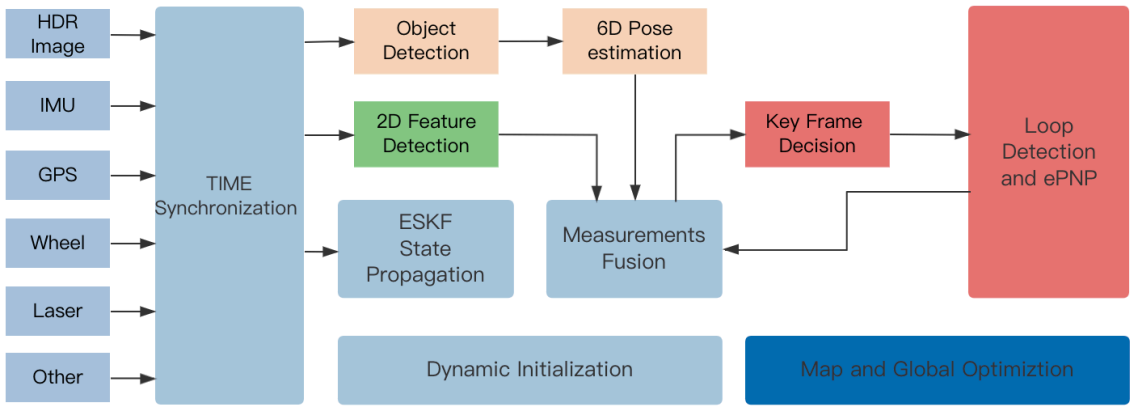
As to the filtering-based method, Jones et al. introduced the EKF framework into the VI-SLAM system. The system realized the estimation of states and parameters as part of the online program and used the EKF framework for effective implementation. Kelly et al. introduced the UKF framework into the VI-SLAM system. This method can calibrate and update the pose online at each update step. MSCKF is the VI-SLAM framework based on the extended Kalman filter proposed by Mourikis in 2007[6]. Only the camera state in the window is added in the state vector, and feature points are ignored, reducing the amount of calculation. The ARL laboratory subsequently launched the Maplab framework with a complete VI-SLAM system with loop detection and relocation functions[7]. The system consists of two parts, one is the online visual inertial global positioning system ROVIOLI (ROVIO with localization integration)[8], which receives the image and inertial sensor data as input, outputs the global pose estimation. The other part is the offline Maplab console, which allows users to

apply various algorithms on the map in offline batch processing. Open VINS is a kind of VINS algorithm based on MSCKF, open-sourced by Huang Guoquan in August 2019[9]. It mainly provides a manifold sliding window Kalman filter, the state consist not only camera intrinsic and extrinsic param eters, but camera-inertial sensor time offset also. The main features of Open VINS are landmark (FEJ) processing with different representations and consistent first estimates, a module for state management such as an integrated system, an extensible visual-inertial system simulator, and a wide range of toolboxes for algorithm evaluation. Its accuracy can be comparable to optimization based SLAM on some open-source data sets.

This paper proposed fusion VIO, a filter-based sensor fusion framework for localization, which provides stable and robust positioning information. The main work discussed in this article contains two parts: 1. FPGA-based high precision time synchronization module. 2. filter-based multi-sensor fusion framework. Firstly, we introduce the overall system structure. At the second part, the FPGA solution can effectively implement the clock synchronization of the sensor data to provide accurate and reliable data to the core algorithm unit. Next, in the fusion part, the open vins framework is extended to implement dynamic system state initialization and multi-sensor fusion. This method effectively solves the error problem caused by the loss of sensor data during the operation of the robot, and improves the robustness of the system.

2. Fusion SLAM - Synchronization

The system architecture of Fusion VIO as shown in Figure 1. The whole SLAM system consists of following six parts:
Sensor data enhancement such as HDR[10,11] and time synchronization (in light blue)
Front-end detection module including 2D image feature detection and object detection (in green and light red)
Loop closure module include key frame decision and loop detection (in dark red)
Sensor fusion module (in light blue)
Mapping module (in dark blue)
Global optimization module (in dark blue)



$$\{q, p, v, ba, bg\}$$

Figure 1. System architecture

2.1 Meaning of synchronization

A well-designed and implemented synchronization module is a must have feature in most of SLAM products, especially a tightly coupled SLAM. The reason is that timing directly determines the real-time characteristics of the system. Real-time can effectively respond to the continuous changes of the real environment and provide reliable and stable location information of the device to the upper-level application. As a result, the upper-level application can quickly respond and make decisions based on the location of the task goal.

2.2 Time synchronization

In order to ensure the real-time performance of the system, Time synchronization accuracy must reach to millisecond level. This accuracy cannot be achieved in a non-real-time operating system since the system is unable to respond in time after the data arrives, and the corresponding time varies with the system's load. We did a few tests on the NVIDIA Jetson TX2 platform, the timing issue mentioned above is 50 ms approximately. The solution to this problem can be divided into two categories: software and hardware synchronization.

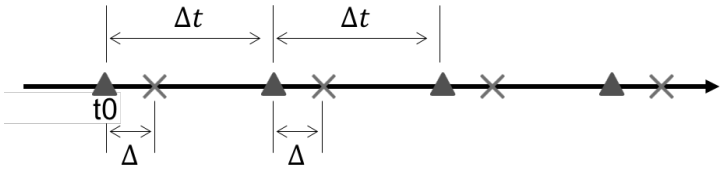


Figure 2. Time synchronization

2.2.1 Software synchronization

In principle, it is feasible to install a real-time patch on a non-real-time system or use a real-time operating system directly. However, this solution requires a lot of time and resources. The development, maintenance, and debugging of real-time operating systems are very different from non-real-time systems such as ubuntu. The difference puts higher requirements on the developer's knowledge and development ability.

2.2.2 Hardware synchronization

Hardware synchronization can be achieved through sensor trigger and FPGA. Sensors with external trigger mechanism are generally too expensive to meet the cost requirements of product. Therefore, our system utilized FPGA for synchronization. All sensors were connected to FPGA and marked with the same standard system timestamp.

Hardware synchronization system architecture is shown in Figure 3.

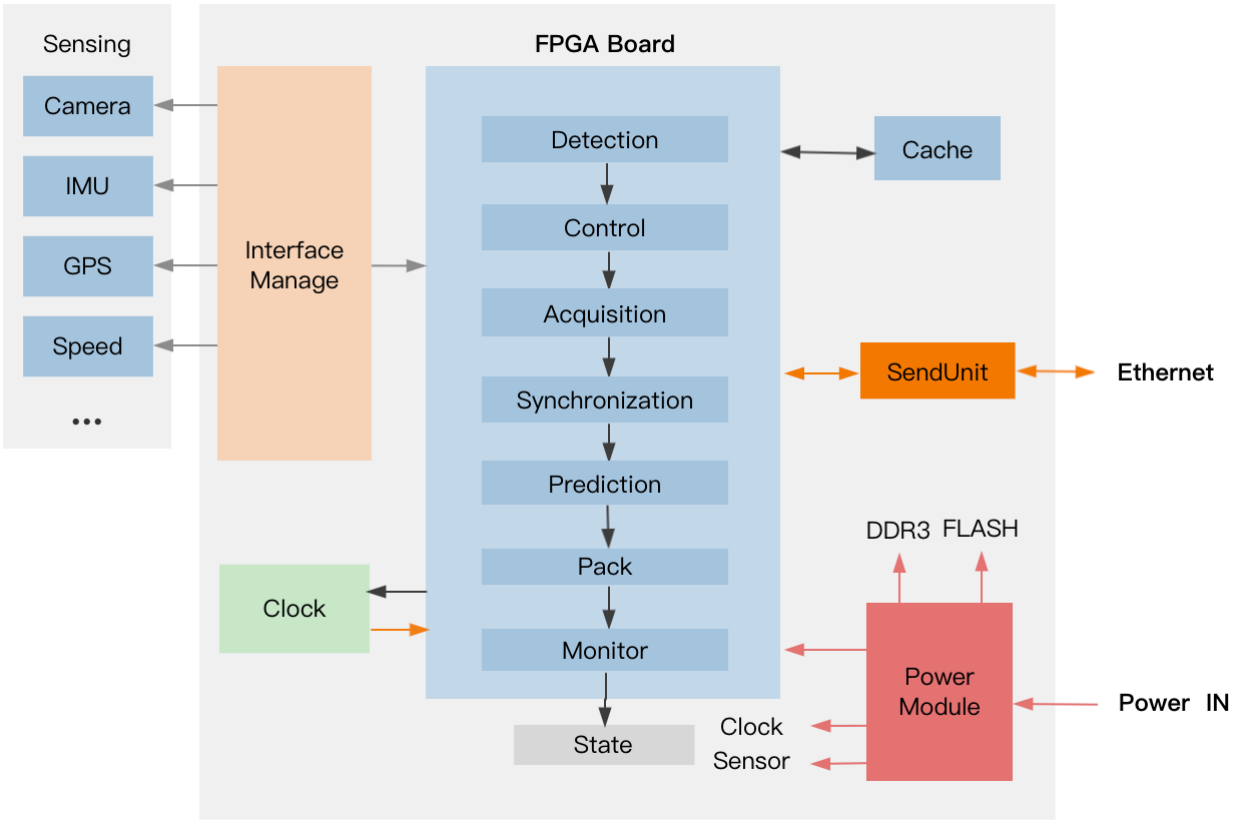


Figure 3. Modules in FPGA architecture

3. Open VINS

The two main contributions of this paper is as follows. Firstly, we extend an EKF based framework of VIO-open vins to fuse wheel odometry and GPS measurements, called fusion VIO. Secondly, we also implemented dynamic initialization of the system.

Open vins is a variant of MSCKF, and its state vector consists of five major parts: $\{ x_I \ x_C \ x_M \ x_W \ c_t \}$, where x_I is $\{ q , p , v , b_a , b_g \}$ represents quaternion, position, velocity, acceleration and gyro bias;

x_C is $\{ q_{ci}, p_{ci} \}$ represents image quaternion and position in the sliding window;
x_M is $\{ q_{mi}, p_{mi} \}$ represents feature quaternion and position in world coordinates;
x_M represents cameras' extrinsic and intrinsic parameters;
c_t represents the time difference between IMU and camera.
The system propagates the statue with IMU measurements by error state Kalman filter[12,13].

3.1 State vector

$$x_k = \begin{bmatrix} x_I^T & x_C^T & x_M^T & x_W^T & c_{tI} \end{bmatrix}^T$$

$$x_I = \begin{bmatrix} {}^{I}_{G^k}q^{-T} & {}^{G}p_{I_k}^T & {}^{G}v_{I_k}^T & b_{w_k}^T & b_{a_k}^T \end{bmatrix}^T$$

$$x_C = \begin{bmatrix} {}^{I}_{G^{k-1}}q^{-T} & {}^{G}p_{I_{k-1}}^T & \cdots & {}^{I}_{G^{k-c}}q^{-T} & {}^{G}p_{I_{k-c}}^T \end{bmatrix}^T$$

$$xM = \begin{bmatrix} {}^{G}p_{f_1}^T & \cdots & {}^{G}p_{f_m}^T \end{bmatrix}^T$$

$$x_W = \begin{bmatrix} {}^{I}_{G_1}q^{-T} & {}^{G_1}p_I^T & \xi_0^T & \cdots & {}^{I}_{G_w}q^{-T} & {}^{G_w}p_I^T & \xi_w^T \end{bmatrix}^T \tag{1}$$

For vector variables, the "boxplus" and "boxminus" operations, which map elements to and from a given manifold, equate to simple addition and subtraction of their vectors:

$$\bar{q}_1 \boxplus \delta\theta \triangleq \begin{bmatrix} \frac{\delta\theta}{2} \\ 1 \end{bmatrix} \otimes \bar{q}_1 \simeq \bar{q}_2 \tag{2}$$

3.2 Motion model

Propagate state with IMU measurements:

$$x_k = f\left(x_{k-1}, {}^{I}a_m, {}^{I}\omega_m, I\right)$$

$$\hat{x}_{k|k-1} = f\left(\hat{x}_{k-1|k-1}, {}^{I}a_m, {}^{I}\omega_m, 0\right) \tag{3}$$

Discrete these function and we get:

$$\hat{x}_{k|k-1} = F\left(\hat{x}_{k-1|k-1}, {}^{I}a_m, {}^{I}\omega_m, 0\right)$$

$$P_{k|k-1} = \Phi_{k-1} P_{k-1|k-1} \Phi_{k-1}^T + Q_{k-1} \tag{4}$$

3.3 Update

Generally, visual updates consist of MSCKF and SLAM feature updates.
MSCKF feature means the visual feature lost in the sliding window.

SLAM feature means the visual feature tracked in the whole window.

$$z_{m,k} = h\left(x_k\right) + n_{m,k}$$

$$zm,k = h\left(\hat{x}_{k|k-1} \boxplus \tilde{x}_{k|k-1}\right) + n_{m,k}$$

$$= h\left(\hat{x}_{k|k-1} + H_k \tilde{x}_{k|k-1}\right) + n_{m,k} \tag{5}$$

$$\Rightarrow \tilde{z}_{m,k} = H_k \tilde{x}_{k|k-1} + n_{m,k} \tag{6}$$

As soon as the lost feature is detected, all the observations will be accumulated and triangulate their global position. There are two parts of Jacobian in residual function, including state Jacobian and feature Jacobian. The second part is composed of features. Because the number of features is different in each frame, a null-left projection must be applied, and then a standard EKF update can be applied as follows:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} \boxplus K_k \left(z_{m,k} - h\left(\hat{x}_{k|k-1}\right)\right)$$

$$P_{k|k} = P_{k|k-1} - K_k H_k P_{k|k-1}$$

$$K_k = P_{k|k-1} H_k^T \left(H_k P_{k|k-1} H_k^T + R_{m,k}\right)^{-1} \tag{7}$$

Except for IMU and feature state, open vins calibrate intrinsic, extrinsic, and time differences between IMU and camera at the same time.

4. Wheel odometry fusion

The main contribution of the paper is that we integrate wheel measurements. Different from visual measurements, the wheel just provides 2D information, such as x, y position and yaw angle. And the three main coordinates frames – body (IMU), camera, and wheel, as shown in Figure 4.

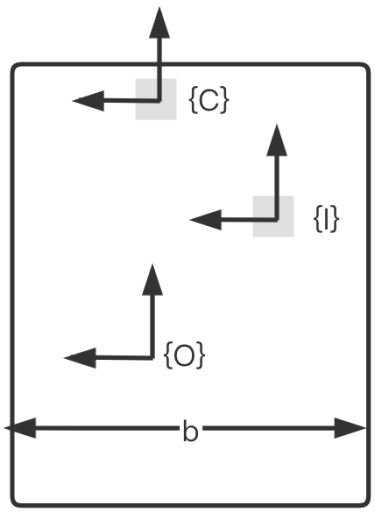


Figure 4. Body and IMU frames

Extend state vector:

$$x_k = \begin{bmatrix} x_{I_k}^T & x_{C_k}^T & x_{WE}^T & O_{t_I} \end{bmatrix}^T \tag{8}$$

where x_{WE}^T is {q,p} represents quaternion and position from wheel to body. O_{t_I} is the time difference between IMU and wheel.

4.1 Wheel measurements per-integration

First, we select all-wheel measurements between two images in that wheel encoder is much quicker than image. We do wheel update right after image update to avoid too much resource consuming.

$$z_{k+1} = \begin{bmatrix} O_{k+1}\dot{\theta} \\ O_k \\ O_k d_{O_{k+1}} \end{bmatrix}$$

$$=: g\left(\omega_{l(k:k+1)}, \omega_{r(k:k+1)}\right) \tag{9}$$

$$z_{k+1} = h\left(x_{I_{k+1}}, x_{C_{k+1}}, x_{WE}, O_{t_I}\right) \tag{10}$$

4.2 Residual and Jacobians

Estimated value residual is equal to per-integrate measurements.

Note that prediction-measurement form is in orientation residual because of the definition of 2D orientation perturbation.

$$z_{k+1} = \begin{bmatrix} O_{k+1}\dot{\theta} \\ O_k \\ O_k d_{O_{k+1}} \end{bmatrix} = \begin{bmatrix} e_3^T \log\left({}^{O}_{I}R^{I_{k+1}}_{G} R^{I_k}_{G} R^T {}^{O}_{I}R^T\right) \\ \Lambda^{O}_{I} R^{I_k}_{G} R\left({}^{G}pI_{k+1} + {}^{I_{k+1}}_{G}R^{TI}\, pO - {}^{G}pI_k - {}^{I_k}_{G}R^{TI}\, pO\right) \end{bmatrix}$$

$$r_\theta = e_3^T \log\left({}^{O}_{I}\hat{R}^{I_{k+1}}_{G} \hat{R}^{I_k}_{G} \hat{R}^T {}^{O}_{I}\hat{R}^T\right) - {}^{O_{k+1}}_{O_k}\theta$$

$$r_d = {}^{O_k}d_{O_{k+1}} - \Lambda^{O}_{I}\hat{R}^{I_k}_{G}\hat{R}\left({}^{G}\hat{p}I_{k+1} + {}^{I_{k+1}}_{G}\hat{R}^{TI}\hat{p}O - {}^{G}\hat{p}I_k - {}^{I_k}_{G}\hat{R}^{TI}\hat{p}O\right) \tag{11}$$

Moreover, compute Jacobians with respect to system states:

$$\frac{\partial \mathrm{h}}{\partial \tilde{x}_{I_{k+1}}} = \begin{bmatrix} e_3^T {}^{O}_{I}\hat{R} & O_{1\times 3} & O_{1\times 9} \\ -\Lambda^{O}_{I}\hat{R}^{I_k}_{G}\hat{R}^{I_{k+1}}_{G}\hat{R}^T \begin{bmatrix} {}^{I}\hat{p}O \end{bmatrix} & \Lambda^{O}_{I}\hat{R}^{I_k}_{G}\hat{R} & O_{2\times 9} \end{bmatrix}$$

$$\frac{\partial \mathrm{h}}{\partial \tilde{x}_{C_{k+1}}}=\begin{bmatrix} -e_3^T\ {}^{O}_{I}\hat{R}_G^{I_{k+1}}\hat{R}_G^{I_k}\hat{R}^T & O_{1\times 3} \\ \Lambda_I^O\hat{R}\left[{}^{I_k}_{G}\hat{R}\left({}^{G}\hat{p}I_{k+1}+{}^{I_{k+1}}_{G}\hat{R}^{TI}\hat{p}O-{}^{G}\hat{p}I_k\right)\right] & -\Lambda_I^O\hat{R}_G^{I_k}\hat{R} \end{bmatrix}$$

$$\frac{\partial \mathrm{h}}{\partial \tilde{x}_{WE}}=\begin{bmatrix} -e_3^T\left(I-{}^{O}_{I}\hat{R}_G^{I_{k+1}}\hat{R}_G^{I_k}\hat{R}^T\,{}^{O}_{I}\hat{R}^T\right) & O_{1\times 3} \\ H_{WE1} & -\Lambda\left(I-{}^{O}_{I}\hat{R}_G^{I_k}\hat{R}_G^{I_{k+1}}\hat{R}^T\,{}^{O}_{I}\hat{R}^T\right) \end{bmatrix}$$

$$H_{WE1}=\Lambda\left(\left[{}^{O}_{I}\hat{R}_G^{I_k}\hat{R}\left({}^{G}\hat{p}I_{k+1}+{}^{I_{k+1}}_{G}\hat{R}^{TI}\hat{p}O-{}^{G}\hat{p}I_k\right)\right]+{}^{O}_{I}\hat{R}_G^{I_k}\hat{R}_G^{I_{k+1}}\hat{R}^T\,{}^{O}_{I}\hat{R}^T\left[{}^{O}\hat{p}I\right]\right) \tag{12}$$

4.3 Update

And then, we can perform a standard EKF update similar to visual updates:

$$z_{k+1}:=g\left(\omega_{l(k:k+1)},\omega_{r(k:k+1)},x_{WI}\right)=h\left(x_{I_{k+1}},x_{C_{k+1}},x_{WE},{}^{O}t_I\right)$$

$$\approx g\left(\omega_{ml(k:k+1)},\omega_{mr(k:k+1)},\hat{x}_{WI}\right)+\frac{\partial g}{\partial \hat{x}_{WI}}\hat{x}_{WI}+\frac{\partial g}{\partial n_\omega}n_\omega$$

$$\approx h\left(\hat{x}_{I_{k+1}},\hat{x}_{C_{k+1}},\hat{x}_{WE},{}^{O}t_I\right)+\frac{\partial h}{\partial \hat{x}_{I_{k+1}}}\hat{x}_{I_{k+1}}+\frac{\partial h}{\partial \hat{x}_{C_{k+1}}}\hat{x}_{C_{k+1}}+\frac{\partial h}{\partial \hat{x}_{WE}}\hat{x}_{WE}+\frac{\partial h}{\partial\,{}^{O}\tilde{t}_I}\,{}^{O}\tilde{t}_I \tag{13}$$

then

$$\tilde{z}_{k+1}:=g\left(\omega_{ml(k:k+1)},\omega_{mr(k:k+1)},\hat{x}_{WI}\right)-h\left(\hat{x}_{I_{k+1}},\hat{x}_{C_{k+1}},\hat{x}_{WE},{}^{O}\hat{t}_I\right)$$

$$\approx\underbrace{\begin{bmatrix}\frac{\partial h}{\partial \hat{x}_{I_{k+1}}} & \frac{\partial h}{\partial \hat{x}_{C_{k+1}}} & \frac{\partial h}{\partial \hat{x}_{WE}} & \frac{\partial h}{\partial \hat{x}_{WI}} & \frac{\partial h}{\partial\,{}^{O}\tilde{t}_I}\end{bmatrix}}_{H_{k+1}}\tilde{x}_{k+1}-\frac{\partial g}{\partial n_\omega}n_\omega \tag{14}$$

Similar to wheel measurements updates, GPS can provide similar data such as velocity to wheel measurements. The main difference is that GPS provides 3D velocity. So in equations (13-14), residual and Jacobians' definitions should extend to 3D.

5. Initialization

Another significant contribution of this paper is an initialization similar to vins-mono[14] and ORB SLAM 3. Dynamic initialization is an important feature of the system.

Usually, EKF based SLAM system is initialized statically. Since it is impossible to stay statically in initialization procedure in some scenarios, it is unsuitable for uncrewed vehicles, such as UAVs.

Except for the state variable X_i, another two parameters have to be initialized in the Initialization procedure. There are s and G_{C0}. s represents visual scale in that the system is equipped with a single camera only. G_{C0} represents the rotation from absolute gravity [0,0,9.81] to IMU of the first frame.

5.1 Static initialization

If the system can be initialized statically, the scale variable is unobservable.

G_{C0} can be calculated from the average of IMU measurements because three axes are perpendicular, and the z-axis is observable. After we gain G_{C0}, remove project of gravity on three axes and ba, bg is determined. Quaternion are set to identity. And position, ba,bg set to zero usually.

5.2 Dynamic initialization

Vision and inertial only pose estimation: Firstly, perform visual, and IMU poses estimation separately.

Visual only initialization is similar to VINS-mono and ORB SLAM 3. The main difference is multi-view triangulation in fusion VIO and two view reconstruction in vins-mono.

IMU pre-integration is as follows:

$$\begin{aligned} p_{b_{k+1}}^{\omega} &= p_{b_k}^{\omega} + v_{b_k}^{\omega}\Delta t_k + \iint_{t\in[k,k+1]}\left(R_t^{\omega}\left(\hat{a}_t - b_{a_t}\right) - g^{\omega}\right)dt^2 \\ v_{b_{k+1}}^{\omega} &= v_{b_k}^{\omega} + \int_{t\in[k,k+1]}\left(R_t^{\omega}\left(\hat{a}_t - b_{a_t}\right) - g^{\omega}\right)dt \\ q_{b_{k+1}}^{\omega} &= q_{b_k}^{\omega} \otimes \int_{t\in[k,k+1]}\frac{1}{2}\Omega\left(\hat{\omega}_t - b_{\omega t}\right)q_t^{b_k}dt \end{aligned} \tag{15}$$

where

$$\Omega(\omega) = \begin{bmatrix} -\lfloor\omega\rfloor_{\times} & \omega \\ -\omega^T & 0 \end{bmatrix}, \lfloor\omega\rfloor_{\times} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{16}$$

The integration method above requires frame bk's rotation, position, and velocity. If starting states change, we need to re-propagate all IMU measurements, which is costly. To avoid re-integration, we adopt a pre-integration algorithm.

$$\begin{aligned} R_{\omega}^{b_k} p_{b_{k+1}}^{\omega} &= R_{\omega}^{b_k}\left(p_{b_k}^{\omega} + v_{b_k}^{\omega}\Delta t_k - \frac{1}{2}g^{\omega}\Delta t_k^2\right) + \alpha_{b_{k+1}}^{b_k} \\ R_{\omega}^{b_k} v_{b_{k+1}}^{\omega} &= R_{\omega}^{b_k}\left(v_{b_k}^{\omega} - g^{\omega}\Delta t_k\right) + \beta_{b_{k+1}}^{b_k} \\ q_{\omega}^{b_k} \otimes q_{b_{k+1}}^{\omega} &= \gamma_{b_{k+1}}^{b_k} \end{aligned} \tag{17}$$

where

$$\begin{aligned} \alpha_{b_{k+1}}^{b_k} &= \iint_{t\in[k,k+1]} R_t^{b_k}\left(\hat{a}_t - b_{a_t}\right)dt^2 \\ \beta_{b_{k+1}}^{b_k} &= \iint_{t\in[k,k+1]} R_t^{b_k}\left(\hat{a}_t - b_{a_t}\right)dt \\ \gamma_{b_{k+1}}^{b_k} &= \iint_{t\in[k,k+1]}\frac{1}{2}\Omega\left(\hat{\omega}_t - b_{\omega_t}\right)\gamma_t^{b_k}dt \\ \hat{\alpha}_{i+1}^{b_k} &= \hat{\alpha}_i^{b_k} + \hat{\beta}_i^{b_k}\delta t + \frac{1}{2}R\left(\hat{\gamma}_i^{b_k}\right)\left(\hat{a}_i - b_{a_i}\right)\delta t^2 \\ \hat{\beta}_{i+1}^{b_k} &= \hat{\beta}_i^{b_k} + R\left(\hat{\gamma}_i^{b_k}\right)\left(\hat{a}_i - b_{a_i}\right)\delta t \\ \hat{\gamma}_{i+1}^{b_k} &= \hat{\gamma}_i^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\left(\hat{\omega}_i - b_{\omega_i}\right)\delta t \end{bmatrix} \end{aligned} \tag{18}$$

then

$$\dot{b}_{a_t} = n_{b_a}, \dot{b}_{\omega_t} = n_{b_\omega}$$

$$\gamma_t^{b_k} \approx \hat{\gamma}_t^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta_t^{b_k} \end{bmatrix} \tag{19}$$

$$\begin{bmatrix} \delta\dot{\alpha}_t^{b_k} \\ \delta\dot{\beta}_t^{b_k} \\ \delta\dot{\theta}_t^{b_k} \\ \delta\dot{b}_{a_t} \\ \delta\dot{b}_{a_t} \end{bmatrix} = \begin{bmatrix} 0 & I & 0 & 0 & 0 \\ 0 & 0 & -R_t^{b_k}\left\lfloor \hat{a}_t - b_{a_t} \right\rfloor_\times & -R_t^{b_k} & 0 \\ 0 & 0 & -\left\lfloor \hat{\omega}_t \right\rfloor & 0 & -I \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta\alpha_t^{b_k} \\ \delta\beta_t^{b_k} \\ \delta\theta_t^{b_k} \\ \delta b_{a_t} \\ \delta b_{\omega_t} \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 & 0 & 0 \\ -R_t^{b_k} & 0 & 0 & 0 \\ 0 & -I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} n_a \\ n_\omega \\ n_{b_a} \\ n_{b_\omega} \end{bmatrix} = F_t \delta z_t^{b_k} + G_t n_t \tag{20}$$

5.3 Gyroscope bias estimation

The difference between two rotations calculated from IMU and visual can be affected by bg.

$$q_{b_k}^{C_0} = q_{b_k}^{C_0} \otimes \left(q_c^b\right)^{-1}$$
$$s\overline{p}_{b_k}^{C_0} = s\overline{p}_{b_k}^{C_0} - R_{b_k}^{C_0} p_c^b$$

$$\min_{\delta b_\omega} \sum_{k \in B} \left\| q_{b_{k+1}}^{C_0\ -1} \otimes q_{b_k}^{C_0} \otimes \gamma_{b_{k+1}}^{b_k} \right\|^2$$
$$\gamma_{b_{k+1}}^{b_k} \approx \begin{bmatrix} 1 \\ \frac{1}{2}\theta_{b_\omega}^{\gamma}\delta b_\omega \end{bmatrix} \otimes \hat{\gamma}_{b_{k+1}}^{b_k} \tag{21}$$

5.4 State estimation

After bg estimated, re-integrated frame pose with estimated bg in the last step. Other state X_I can be calculated as follows:

$$x_I = \left[v_{b_0}^{b_0}, v_{b_1}^{b_1}, \cdots, v_{b_n}^{b_n}, g^{C_0}, s \right],$$
$$\hat{\alpha}_{b_{k+1}}^{b_k} = R_{c_0}^{b_k}\left(s\left(\overline{p}_{b_{k+1}}^{c_0} - \overline{p}_{b_k}^{c_0}\right) + \frac{1}{2} g^{c_0} \Delta t_k^2 - R_{b_k}^{c_0} v_{b_k}^{b_k} \Delta t_k \right)$$
$$\hat{\beta}_{b_{k+1}}^{b_k} = R_{c_0}^{b_k}\left(R_{b_{k+1}}^{c_0} v_{b_{k+1}}^{b_{k+1}} + g^{c_0}\Delta t_k - R_{b_k}^{c_0} v_{b_k}^{b_k} \right) \tag{22}$$

$$\hat{z}_{b_{k+1}}^{b_k} = \begin{bmatrix} \hat{\alpha}_{b_{k+1}}^{b_k} - p_c^b + R_{c_0}^{b_k} R_{b_{k+1}}^{c_0} p_c^b \\ \hat{\beta}_{b_{k+1}}^{b_k} \end{bmatrix} = H_{b_{k+1}}^{b_k} x_I + n_{b_{k+1}}^{b_k} \approx \begin{bmatrix} -I\Delta t_k & 0 & \frac{1}{2} R_{c_0}^{b_k} \Delta t_k^2 & R_{c_0}^{b_k} \left(\overline{p}_{c_{k+1}}^{c_0} - \overline{p}_{c_k}^{c_0} \right) \\ -I & R_{c_0}^{b_k} R_{b_{k+1}}^{c_0} & R_{c_0}^{b_k} \Delta t_k & 0 \end{bmatrix} \begin{bmatrix} v_{b_k}^{b_k} \\ v_{b_{k+1}}^{b_{k+1}} \\ g^{c_0} \\ s \end{bmatrix}$$

$$\min_{x_I} \sum_{k \in \mathrm{B}} \left\| \hat{z}_{b_{k+1}}^{b_k} - H_{b_{k+1}}^{b_k} x_I \right\|^2 \tag{23}$$

Table 1. Mono comparison

		V101				V102				V103			
		Time	CPU	Memory	Accuracy	Time	CPU	Memory	Accuracy	Time	CPU	Memory	Accuracy
VINS Fusion	max	150	113.30	83228.00	2.64	97	120.00	79916.00	2.11	113	113.30	81088.00	2.51
	avg		100.56	75986.67			100.00	74428.92			82.75	75151.20	
OPEN VINS	max	49	100.00	96924.00	2.64	30	100.00	89980.00	2.60	35	106.70	93072.00	2.13
	avg		98.33	93727.00			96.50	89716.00			102.23	90993.33	
ORB SLAM3	max	160	246.70	760396.00	4.38	97	226.70	711388.00	3.65	118	206.70	750548.00	3.66
	avg		188.15	623451.75			184.00	586261.60			180.57	614971.67	
Ours	max	48	101.30	97112.00	2.65	32	110.00	79932.00	2.61	37	108.70	98781.00	2.14
	avg		98.00	89781.50			95.50	75253.00			101.30	95235.00	
		V201				V202				V203			
		Time	CPU	Memory	Accuracy	Time	CPU	Memory	Accuracy	Time	CPU	Memory	Accuracy
VINS Fusion	max	117	106.70	79580.00	5.34	121	113.30	71804.00	X	120	126.70	74356.00	4.34
	avg		92.01	75460.80			81.92	70695.27			100.61	71058.55	
OPEN VINS	max	34	106.70	89376.00	5.20	38	100.00	89072.00	4.54	29	100.00	89416.00	4.23
	avg		100.00	88006.67			97.93	87346.67			100.00	88220.00	
ORB SLAM3	max	127	220.00	752224.00	3.52	130	226.70	754184.00	3.63	128	213.30	770780.00	3.71
	avg		175.06	601584.31			191.38	623500.00			174.36	614313.23	
Ours	max	34	108.60	91789.00	5.30	37	100.00	91002.00	4.51	28	100.00	89523.00	3.90
	avg		101.30	89456.00			97.20	89231.00			99.10	88965.00	

Table 2. Stereo comparison

		V101				V102				V103			
		Time	CPU	Memory	Accuracy	Time	CPU	Memory	Accuracy	Time	CPU	Memory	Accuracy
VINS Fusion	max	150	146.70	89748.00	2.47	97	180.00	84076.00	2.40	113	180.00	82908.00	2.36
	avg		128.50	83526.57			129.90	80057.50			115.60	78573.45	
OPEN VINS	max	88	106.70	104436.00	2.68	51	106.70	97828.00	2.53	62	106.70	101024.00	2.41
	avg		100.90	102314.50			101.34	95874.40			101.34	100668.80	
ORB SLAM3	max	169	400.00	808764.00	4.38	102	333.30	760744.00	4.27	123	380.00	835072.00	3.76
	avg		304.16	662796.25			262.08	609962.00			284.68	646493.00	
Ours	max	89	105.20	114632.00	2.69	50	105.66	102350.00	2.53	66	110.00	112356.00	2.51
	avg		100.30	102346.00			100.50	99456.00			105.00	109988.00	
		V201				V202				V203			
		Time	CPU	Memory	Accuracy	Time	CPU	Memory	Accuracy	Time	CPU	Memory	Accuracy
VINS Fusion	max	117	173.30	89928.00	5.12	120	160.00	83432.00	4.52	120	146.70	81152.00	4.21
	avg		124.85	84225.45			131.53	78533.45			117.55	76656.00	
OPEN VINS	max	64	113.30	96732.00	5.19	68	106.70	122304.00	4.49	53	100.00	99476.00	4.13
	avg		104.53	95302.67			101.12	119480.00			97.42	98039.20	
ORB SLAM3	max	134	320.00	744232.00	3.57	136	393.30	855888.00	3.73	132	426.70	931184.00	3.69
	avg		265.03	603560.92			287.14	676458.00			282.02	704539.38	
Ours	max	61	109.00	95879.00	4.80	71	326.70	113253.00	4.20	52	99.00	98465.00	4.30
	avg		100.00	93546.00			180.00	109980.00			95.30	94651.00	

6. Experience results

We evaluate the proposed fusion VIO with datasets recorded in the lab, which equipped a motion capture system.

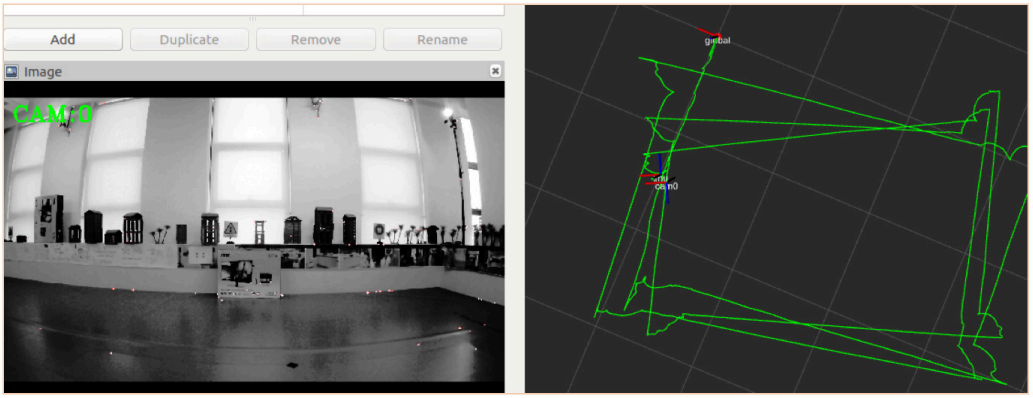


Figure 5. The environment of experience

The following configuration achieved the experiments:
Mono and Stereo camera
Low-cost imu
X86 platform (Intel i5 Quad Core CPU, 8G DDR4 memory)

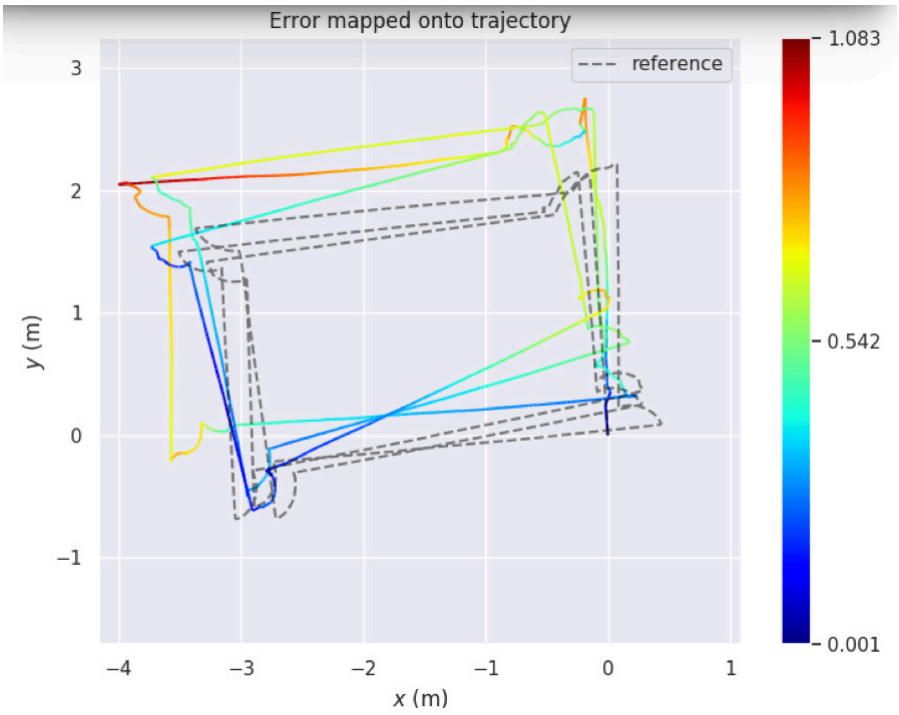


Figure 6. Difference solution in the indoor scene

The experiments prove that the fusion VIO achieved similar accuracy but cost one-third of resources compared to ORB SLAM 3.

7. Conclusion and future work

In this paper, we have presented our Fusion VIO system as a platform for industrial purposes. We provide an EKF based visual SLAM framework to fuse GPS or wheel encoder measurements. The method stated above can fuse any sensor

provides speed measurements. In particular, we proposed methods of dynamic initialization.

We plan to expand our system to provide the ability to fuse loop closure information in the future. We believe that this feature makes the system more robust. We are also interested in integrating visual-inertial mapping and semantic perception capabilities into Fusion SLAM.

References

[1] Bloesch M, Burri M, Omari S, et al. Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback[J]. *International Journal of Robotics Research*, 2017, 36(10): 1053-1072.

[2] Faessler M, Fontana F, Forster C, et al. Autonomous, vision based flight and live dense 3D mapping with a quadrotor microaerial vehicle[J]. *Journal of Field Robotics*, 2016, 33(4): 431-450.

[3] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart and Paul Timothy Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. The International Journal of Robotics Research, 2015.

[4] S. Leutenegger, M. Chli and R. Y. Siegwart, "BRISK: Binary Robust invariant scalable keypoints," 2011 International Conference on Computer Vision, 2011, pp. 2548-2555, doi: 10.1109/ICCV.2011.6126542.

[5] Campos C, Elvira R, Juan J. Gómez Rodríguez, et al. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM[J]. 2020.

[6] A multi-state constraint Kalman filter for vision-aided inertial navigation[C] IEEE International Conference on Robotics and Automation. Piscataway, USA:IEEE, 2007: 3565-3572.

[7] T. Schneider and M. T. Dymczyk and M. Fehr and K. Egger and S. Lynen and I. Gilitschenski and R. Siegwart. Maplab: An Open Framework for Research in Visual-inertial Mapping and Localization. IEEE Robotics and Automation Letters, 2018.

[8] Bloesch, Michael; Omari, Sammy; Hutter, Marco; Siegwart, Roland, ROVIOLI - Robust Visual Inertial Odometry with Localization Integration, 2015.

[9] Geneva P, Eckenhoff K, Lee W, et al. OpenVINS: A Research Platform for Visual-Inertial Estimation[C] Proc. of the IEEE International Conference on Robotics and Automation. IEEE, 2020.

[10] S. Q. Wu, Z. G. Li, J. H. Zheng, and Z. J. Zhu, Exposure-Robust Alignment of Differently Exposed Images, IEEE Signal Processing Letters, Vol. 21, No. 7, pp. 885-889, Jul. 2014.

[11] J. Jiang, Z. G. Li, S. L. Xie, S. Q. Wu, and L. C. Zeng, Robust Alignment of Multi-Exposed Images With Saturated Regions, IEEE Access, Vol. 8, pp. 221689-221699, Dec. 2020.

[12] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, On-Manifold Preintegration for Real-Time Visual-Inertial Odometry, IEEE Trans. Robot., vol. 33, no. 1, Feb. 2016.

[13] J. Henawy, Z. G. Li, W. Y. Yau, and G. Seet, Accurate IMU Factor Using Switched Linear Systems for VIO, On Industrial Electronics, Vol. 68, No. 8, pp. 7199-7208, Aug. 2021.

[14] Tong Qin, Peiliang Li, Zhenfei Yang, Shaojie Shen, VINS-Mono A Robust and Versatile Monocular Visual-Inertial State Estimator, 2017.