sensors

MDPI

Article

Autonomous Dam Surveillance Robot System Based on Multi-Sensor Fusion †

Chao Zhang [1], Quanzhong Zhan [1], Qi Wang [2,*], Haichao Wu [2], Ting He [1] and Yi An [3]

[1] Information Center of the Ministry of water resources of the P.R.C, Beijing 100053, China; zhangchao@mwr.gov.cn (C.Z.); zcqcz@mwr.gov.cn (Q.Z.); heting@mwr.gov.cn (T.H.)
[2] Beijing Tritalent Intelligence Technology Co., Ltd., Beijing 100078, China; wuhaichao@tritalent.cn
[3] School of Control Science and Engineering, Dalian university of technology, Dalian 116024, China; anyi@dlut.edu.cn
* Correspondence: wangqi@tritalent.cn
† This paper is an extended version of our paper published in Zhang, C.; Wang, Q.; Zhan, Q.; He, T.; An, Y. Autonomous System Design for Dam Surveillance Robots. In Proceedings of the IEEE Smart World Congress, Leicester, UK, 19–23 August 2019.

Received: 31 December 2019; Accepted: 12 February 2020; Published: 17 February 2020

Abstract: Dams are important engineering facilities in the water conservancy industry. They have many functions, such as flood control, electric power generation, irrigation, water supply, shipping, etc. Therefore, their long-term safety is crucial to operational stability. Because of the complexity of the dam environment, robots with various kinds of sensors are a good choice to replace humans to perform a surveillance job. In this paper, an autonomous system design is proposed for dam ground surveillance robots, which includes general solution, electromechanical layout, sensors scheme, and navigation method. A strong and agile skid-steered mobile robot body platform is designed and created, which can be controlled accurately based on an MCU and an onboard IMU. A novel low-cost LiDAR is adopted for odometry estimation. To realize more robust localization results, two Kalman filter loops are used with the robot kinematic model to fuse wheel encoder, IMU, LiDAR odometry, and a low-cost GNSS receiver data. Besides, a recognition network based on YOLO v3 is deployed to realize real-time recognition of cracks and people during surveillance. As a system, by connecting the robot, the cloud server and the users with IOT technology, the proposed solution could be more robust and practical.

Keywords: water conversancy robot; dam surveillance; autonomous navigation; fusion localization

1. Introduction

In the water conservancy industry, there are many fundamental engineering facilities, such as dams, water and soil conservation, water transfer project, shipping project, water supply project, hydraulic power plants, irrigation facilities, etc. Big water dams are the most comprehensive facilities. They always have many functions, such as flood control, electric power generation, irrigation, water supply, shipping, etc. [1]. Therefore, the safety of big dams is crucial to cities and people around.

In the past, staff must check the environment, structure, and electromechanical facilities of dams regularly every day. This is a necessary way of keeping the dam running safely. However, it cannot realize all-weather all-day monitor and sometimes staffs are at risk since most dams are constructed in remote rural areas. Nowadays, with the great improvement of robot technology, robots have been used in public safety, security check, disaster rescue, and high-voltage lines inspection [2–5]. Many researchers and engineers are also paying attention to utilize underwater robots, unmanned aerial vehicles, unmanned surface vehicles for dams' surveillance and inspection.

However, applications of ground mobile robots are very rare. It is not easy for a ground robot to do the dam surveillance job autonomously. Firstly, the working places for most dams are off-road terrains, which needs a small, powerful, and agile robot mechanics design. Secondly, the GNSS satellite signal cannot fully cover all working area. The robot is difficult to continue path planning without accurate position information. Besides, to make a comprehensive check for a dam, the robot needs carry kinds of sensors. The robot must be capable of real-time data analyzing. In addition to the above, the robot system has to be low cost and robust.

In the autonomous robot area, simultaneous localization and mapping (SLAM) technology has been applied in many practical robot applications, especially laser SLAM. Researchers have proposed many laser-SLAM algorithms to build 2D/3D maps and realize localization [6,7]. Visual odometer (VO) is another hot topic in this area [8,9]. Cameras are relatively cheap and visual algorithms could show better performance in some environments. Many path planning methods are also being proposed to solve the obstacle avoiding problem [10,11]. The navigation technology now is ready for mobile robot design. Developing an autonomous dam surveillance robot system presents a wide variety of challenges. The robot should move on all kinds of ground surfaces and sometimes should move on narrow spaces. GNSS signal cannot cover all the work areas and it is difficult to acquire an accurate position. Online real-time inspection should be applied to the robot and the inspection algorithms should work with small computing resources.

This paper presents a general system for the mobile robot, which includes the robot, the cloud server, and terminals. The system can watch surveillance jobs status and can control the robot remotely when an emergency. As the robot body itself, we design a small powerful wheeled skid-steering mobile platform. Compared to alternative wheel configurations such as Ackerman or axle-articulated, the skid-steering platform shows two major advantages. First, it is simple and robust in mechanical terms. Second, it provides better maneuverability, including zero-radius turning. After analyzing the kinematic model of the skid-steered robot, we can estimate wheel odometry well with an onboard IMU (inertial measurement unit).

In this paper, a low-cost LiDAR sensor is used for laser odometry. This Livox MID40 LiDAR is a high-performance low-cost LiDAR sensor easily available. However, the LiDAR only has a 38.4° FOV and has a novel non-repeat scan mode, which differs a lot with common laser rangefinders. The method LiDAR odometry and mapping (LOAM) is adopted here for odometry estimation from its point cloud data. The method here avoids high computing burden since we only use the data for odometry calculation, not for real-time mapping. Besides, the robot was equipped with a GNSS receiver and an IMU. Therefore, we have position data from different sources: the LiDAR odometry, the longitude and latitude data, the wheel odometry and IMU measured data. To build a robust and fast localization system, we designed two extended Kalman filter (EKF) nodes. The first EKF node fuses IMU data, wheel odometry, LiDAR odometry to get robot transform in the inertial frame. By running it on the robot STM32 MCU board, we can ensure the fusion has real-time performance with high frequency. The second fuses results of first node, IMU data, longitude and latitude data to get robot transform in the world frame. The second node is running on the upper PC. The longitude and latitude data would be the observer position when the GNSS signal is good. After the position is acquired, we use a dynamic window approach (DWA) algorithm to do motion planning.

The paper is organized as follows. Some related work on robot control, fusion localization, object detection methods is summarized in Section 2. The general robot design is shown in Section 3. The mechanical design and control of the mobile robot can be seen in Section 4. All mounted sensors would be introduced in Section 5. Localization solution and some practical results could be found in Section 6. The online detection method is described in Section 7. Finally, a short conclusion is given in Section 8.

2. Related Work

To solve the dam surveillance and inspecting the problem, many robotics and sensor technologies have been deployed. A GPS-based surveillance robot is proposed in [12] to monitor the exterior

deformation of the dam. A mechanically scanned imaging sonar (MSIS) is used for locating the underwater robot relative position for dam wall surveillance [13]. Many researchers utilized UAVs for dam image collecting and modeling, which is low-cost and convenient [14,15].

The wheeled skid-steering mobile robot is agile and strong, but its kinematics is complex since the drift always occurs and difficult to measure [16]. In [16], an IMU-based EKF fusion method is also proposed to improve the wheel odometry. To improve real-time motion control and dead-reckoning, a method to experimentally obtain an optimized kinematic model of the robot is presented in [17]. A detail model analysis and a back-stepping controller design are proposed in [18], simulation results for trajectory tracking are good. To find the correct model parameters [19], designed analysis and experimental kinematic scheme for skid-steering wheeled robot based-on a laser scanner sensor. The practical experiments show improvement in dead-reckoning performance.

A real-time kinematic (RTK) GPS sensor can help the robot to get an accurate position. It is commonly used in wide-open space [20]. Laser SLAM is now a relatively mature way to do mapping and localization (LiDAR odometry and mapping). LOAM proposed in [21] is one of the best LiDAR SLAM methods. The LOAM was running on the Velodyne Puck LiDAR which has 16 laser lines. Recently, a novel version of LOAM is proposed in [22], which is running on a low-cost novel Livox LiDAR and shows good performance in experiments. For land robots, dead reckoning is another reliable source of odometry. A robust AHRS could improve the wheel dead reckoning well [23].

Localization is one of the most important elements in the robot system. When the GPS/GNSS signal is too weak to use, a framework combines keyframe-based visual-inertial odometry with novel geometric image-based localization is built to provide a real-time estimate of a UAV's pose. Weak GPS feeds are used as a weak prior for suggesting loop closures [24]. A Monte Carlo Localization system that fuses wheel and visual odometry for the prediction are designed for sewer inspection robots. The update step takes into account the network topology and other active methods. The system works well in practical experiments [25]. To address existing positioning systems do not perform well in urban canyons, tunnels, and indoor parking lots for driverless cars, an EKF based multi-sensor positioning system that combines a GPS sensor, a camera, and in-vehicle sensors assisted by kinematic and dynamic vehicle models [26]. Among all the localization system, EKF is widely used and shows good performances [27,28].

For dam surveillance, people and dam crack detection are important to ensure people and dam safety. Vision-based approaches to detect, track and identify people on a mobile robot in real-time has been used for many years. Thermal and grey images are successfully used to do the people recognition job on a mobile security robot [29]. Deep Learning shows great potential for crack damage detection. A deep architecture of convolutional neural networks (CNNs) method is proposed in [30] to find concrete cracks for civil infrastructures.

Therefore, a wheeled skid-steering mobile robot is built for the dam surveillance, which carries an onboard IMU, wheel encoders, a Livox LiDAR sensor, a GNSS antenna, a monocular industrial camera, and a customized MCU controller. We proposed the robot platform controller design, odometry methods, fusion localization, crack detection network to make the whole system low-cost and robust.

3. General Design

In dam surveillance applications, there are three main modules, which are the robot itself, the cloud server and the front-end terminals. The robot is responsible for moving in the workplace, collecting monitor data and handling some online detection applications. The cloud server takes charge of data processing and transferring. The users can watch the robot status, do manual analysis, and control the robot remotely based on real-time video streaming when necessary.

The robot and the server are connected by mobile internet and WIFI. The server and the terminal users are connected by the Internet and LAN. The general structure diagram is shown in Figure 1. The robot itself and the cloud server are connected via message queuing telemetry transport protocol

(MQTT). Robot navigation data can transfer to the server and the server can send job command and global navigation plan to the robot. This paper mainly focuses on the robot module.

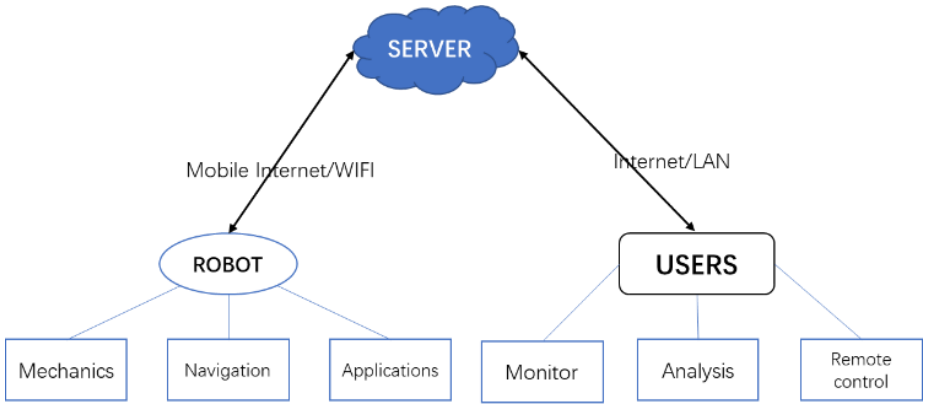

Figure 1. General design scheme for a dam surveillance robot.

A state machine is designed here for robot control, which includes self-check, autonomous navigation, remote control, idle, and emergency states. The robot would get into self-check state once the power is turned on. All sensors, motors, localization data, the high-level computer will then be checked. The robot would get into an idle state and wait for task command from the cloud server if the self-check is passed. Autonomous navigation would be triggered once the task and waypoints are received. The emergency state would be triggered when the robot is stuck or some sensors are running wrong. The logic of the state machine is shown in Figure 2.

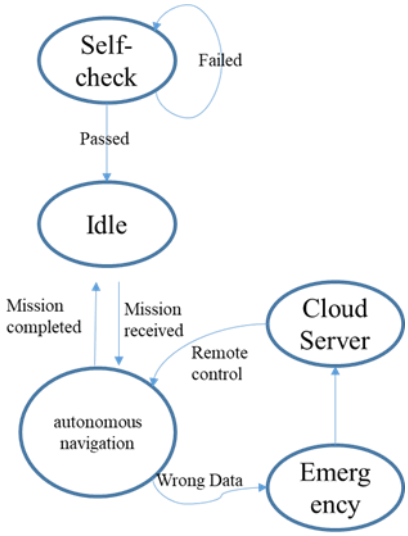

Figure 2. State machine logic structure.

4. Robot Platform

To realize high robustness and agile motion performance, the robot body is built with an all-aluminum chassis which about a total of 10 Kg weight. Two 200W DC brushless high torque motors are deployed to realize four-wheel drive. A 24 V lithium battery with 40 AH capacity is used as the power supply.

All motors are controlled by a customized MCU board, which connects the motors and drivers by CAN bus. The maximal speed is 2 m/s; the maximum payload is 40 Kg; the maximum runtime is more than 4 hours. It is suitable for rugged all-terrain operation with four off-road tires. Figure 3 shows the body design.

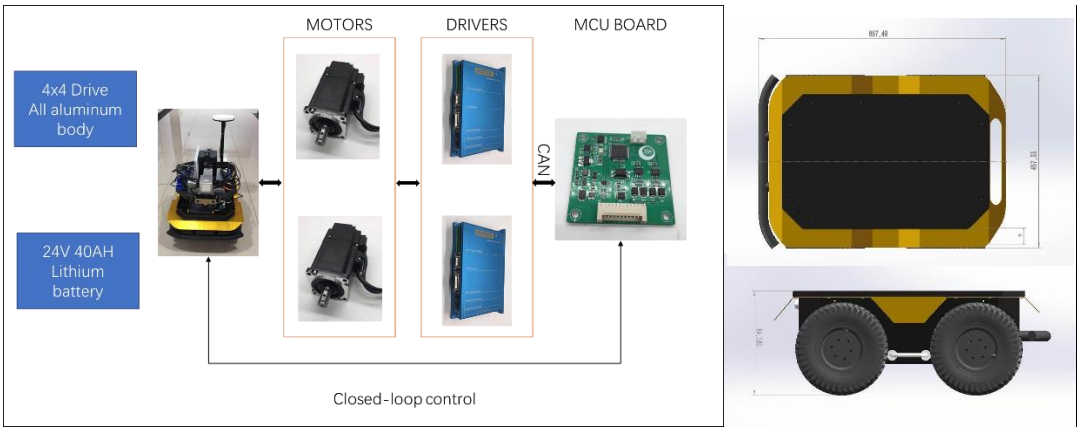

Figure 3. Mechanical layout and onboard control diagram.

As shown in Figure 4, for the robot body, we will use the velocity and the angular velocity in the robot axis as state variables, i.e., $[v_x \; w]^T$. After the kinematic analysis of the robot, we can find the relationship between robot velocities and wheel speed.

$$\begin{bmatrix} v_x \\ w \end{bmatrix} = r\frac{\pi d}{60n}\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2c} & -\frac{1}{2c} \end{bmatrix}\begin{bmatrix} W_l \\ W_r \end{bmatrix} \tag{1}$$

where r is so called effective radius of wheels, n is the reduction ratio, $2c$ is a spacing wheel track.

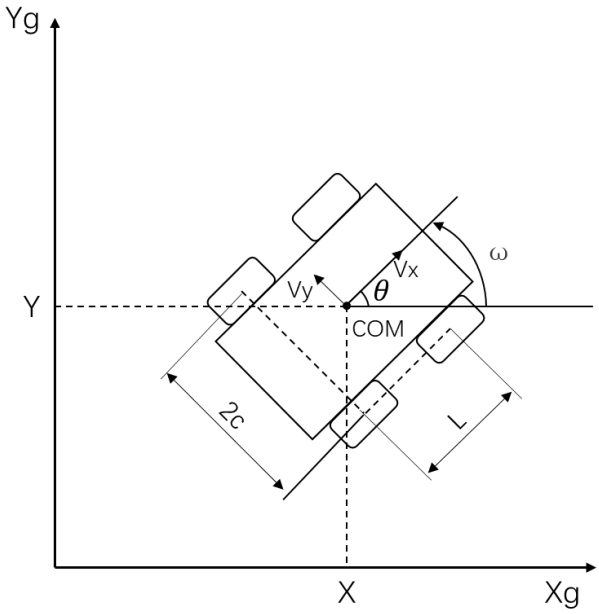

Figure 4. Kinematic model of the robot.

For the robot body, it will get velocity and angular velocity command $\left[v_x^d \; w^d\right]^T$ from upper PC. We should control the velocities to track the command. From (1), we have

$$\begin{cases} W_l + W_r = \frac{2v_x^d}{r\frac{\pi d}{60n}} \\ -W_l + W_r = \frac{2w^d c}{r\frac{\pi d}{60n}} \end{cases} \tag{2}$$

From (2), we could calculate the desired wheel velocities easily, but the direct calculation does not work well since the model is a realistic model and the angular velocity always has a large lag. Therefore, we add an onboard IMU and use the following equation to set the wheel velocities.

$$\begin{cases} v_x = v_x^d \\ w = k_p e_w + k_i \int e_w + k_d \dot{e}_w \end{cases} \tag{3}$$

where k_p, k_i, k_d are constants for the simple PID controller, $e_w = w^d - w^i$, w^i is the feedback from the onboard IMU. The practical response could be seen in Figure 5.

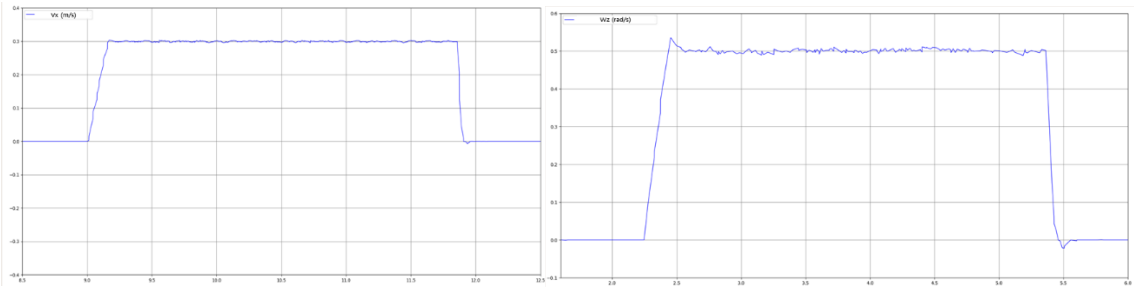

Figure 5. Practical response of the forward velocity and angular velocity.

It is simple to calculate the wheel odometry in the inertial frame. We drive the robot moving a square and back to the origin, the dead reckoning result is shown in Figure 6 and final position value is [0.1229–0.2654 0.03] (should be $[0\ 0\ 0]^{\mathrm{T}}$). The whole distance is about 70 m and we did experiments 10 times, the origin RMSE (root mean square errors) is 0.3135.

$$\begin{cases} \begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \\ \qquad\qquad\qquad\qquad\qquad\qquad \mathrm{Z} \\ \dot{\theta} = w^i \\ v_y = -X_{ICR} \cdot w^i \end{cases} \tag{4}$$

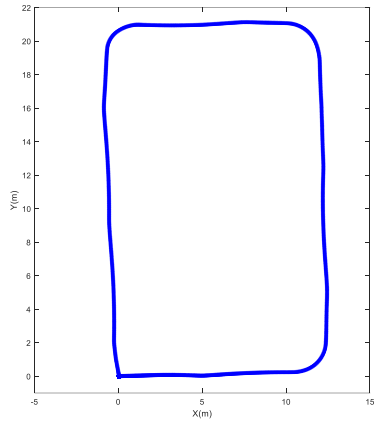

Figure 6. Dead reckoning results.

5. Sensors

The experimental robot with sensors is shown in Figure 7. From the front to the back, there are 1 GNSS receiver, a stereo camera, a Livox LiDAR, a monocular camera, a 4G LTE wireless router, an industrial computer, a GNSS antenna. An onboard MCU controller with IMU is located inside the

body box. Motion control is handled by the onboard MCU which also sends odometry data to the high-level industrial computer. The navigation is handled by the computer which is connected to the cloud server.

Figure 7. Experiment robot with sensors.

A navigation map is built mainly by the LiDAR and odometry data using the LOAM SLAM method. The map would be regarded as prior knowledge. The monocular camera in the middle is used to transfer video streaming and do the people and crack detection jobs. The GNSS receiver with antenna can acquire accurate position data in the wide-open area. On the other hand, the LiDAR's point cloud and cameras' depth data are designed to be used in obstacles perception as well in the future.

6. Localization and Navigation

The navigation includes perception, localization, path planning, and a remote controller. The whole diagram is depicted in Figure 8. The mature cost grid map is used for general environment representation. The filtered point cloud and depth data would be obstacles shown on the map.

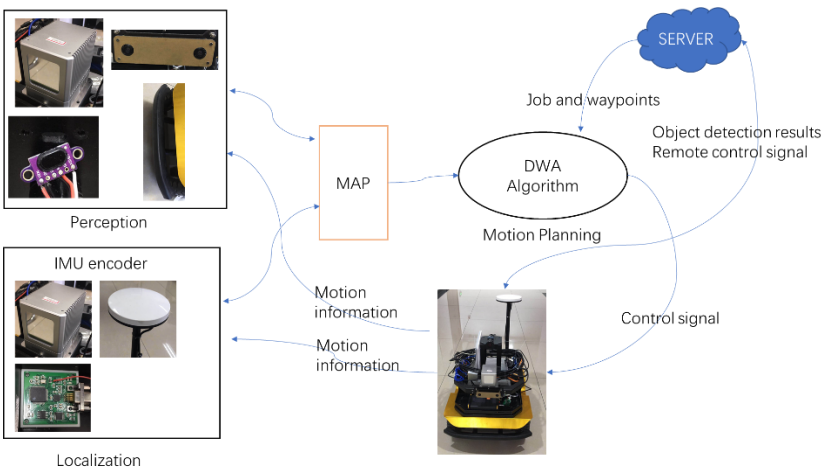

Figure 8. Navigation diagram.

The path planning method is divided into two parts here. The global waypoints are sent to the robot by the cloud server, which is pre-defined by different jobs. The local motion planning is achieved by the DWA algorithm on the grid map.

Localization is one of the most difficult problems for dam robots since the environment is changing and the robot has to move indoor and outdoor. Using one single sensor to acquire a positon is an impossible task. Many sensors would introduce fusion problem. Extended Kalman filter (EKF) is a good way to solve this problem and has been deployed in many robot localization applications.

A two EKF node structure is proposed here to build a robust localization system. The GNSS data is transformed based on the local origin and then input to EKF global node update process. The yaw angle from the LiDAR odometry is input to the EKF update process. The result of the EKF local node is input to the EKF global predict process. The LiDAR odometry data are input to the EKF local update process. The wheel odometry data and the IMU measurement is input to the EKF local predict process. It is noted here that the local EKF node is running on the embedded MCU at 200Hz and it handles in 3D space with 16 states.

It is important to get the 3D pose information for the rubber ground surface. However, the global EKF node is running at 30Hz on the computer and it only produces in 2D space. The robot position and orientation in the map are acquired with the EKF global updating. The fusion localization structure diagram is shown in Figure 9. Where W_l^d, W_l^d are the wheel velocity command, W_l, W_l are the feedback wheel velocity from encoders, V_x^d, ω^d are the velocity command generated from motion planning, V_x, ωz are the wheel odometry outputs, x, y, yaw are the LiDAR odometry, X, Y, Yaw are the final global pose estimation result.

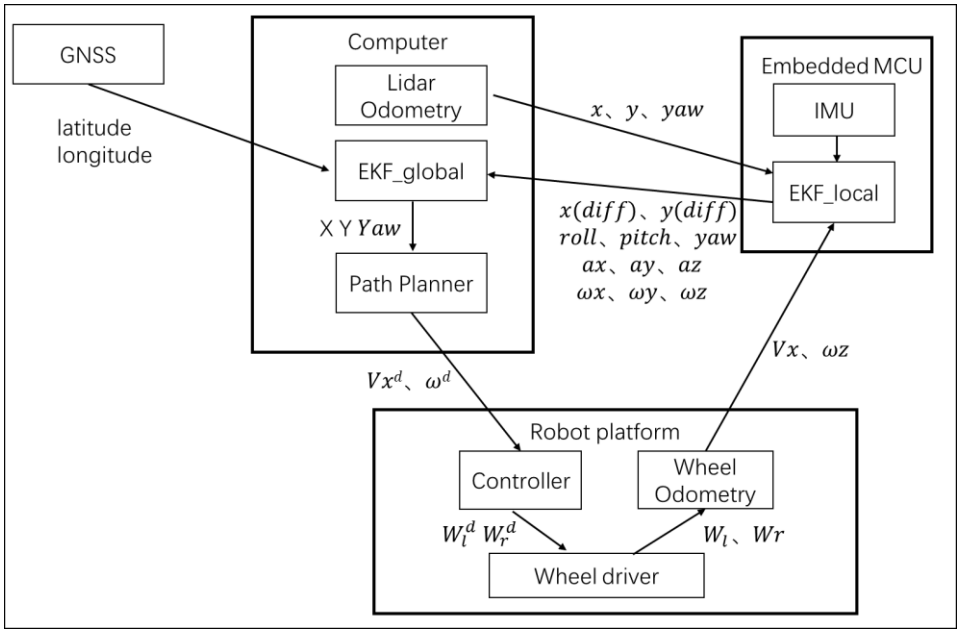

Figure 9. Fusion localization structure diagram.

With the kinematic models and zero-velocity constraints, we are now ready to design a kinematic-model-based robot positioning scheme for the local EKF node. The general state update equation is

$$x_t = Fx_{t-1} + Gu_t \tag{5}$$

where $x_t = [q_0\ q_1\ q_2\ q_3\ x\ y\ z\ V_x\ V_y\ V_z\ \delta\varphi_{bias}\ \delta\theta_{bias}\ \delta\psi_{bias}\ \delta V_{xbias}\ \delta V_{ybias}\ \delta V_{zbias}]^T$ is the state variables, $q_0\ q_1\ q_2\ q_3$ are the quaternion, $x\ y\ z$ are the position in the inertial frame, $V_x\ V_y\ V_z$ are the corresponding

velocities, $\delta\varphi_{bias}$ $\delta\theta_{bias}$ $\delta\psi_{bias}$ are the rotation zero-offset, δV_{xbias} δV_{ybias} δV_{zbias} are the acceleration zero-offset. We have the following detail state update equations.

$$\begin{bmatrix} q_0(k+1) \\ q_1(k+1) \\ q_2(k+1) \\ q_3(k+1) \end{bmatrix} = q(h) \bullet \begin{bmatrix} q_0(k) \\ q_1(k) \\ q_2(k) \\ q_3(k) \end{bmatrix} \tag{6}$$

$$q(h) = \begin{bmatrix} 1 \\ \frac{\delta\phi}{2} - \frac{\delta\phi\ _{bias}}{2} \\ \frac{\delta\theta}{2} - \frac{\delta\theta_{bias}}{2} \\ \frac{\delta\psi}{2} - \frac{\delta\psi_{bias}}{2} \end{bmatrix} \tag{7}$$

where $\delta\phi$, $\delta\theta$, $\delta\psi$ are the radian variations during dt.

$$\begin{bmatrix} V_x(k+1) \\ V_y(k+1) \\ V_z(k+1) \end{bmatrix} = \begin{bmatrix} V_x(k) \\ V_y(k) \\ V_z(k) \end{bmatrix} + \begin{bmatrix} g_x(k) \\ g_y(k) \\ g_z(k) \end{bmatrix} \bullet dt + T_b^n \delta V \tag{8}$$

where $g_x(k) g_y(k) g_z(k)$ are the components of gravity.

$$\delta V = \begin{bmatrix} \delta V_x \\ \delta V_y \\ \delta V_z \end{bmatrix} - \begin{bmatrix} \delta V_{xbias} \\ \delta V_{ybias} \\ \delta V_{zbias} \end{bmatrix} \tag{9}$$

where δV_{xbias}, δV_{ybias}, δV_{zbias} are the velocity variations during dt.

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ z(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ z(k) \end{bmatrix} + \begin{bmatrix} V_x(k) \\ V_y(k) \\ V_z(k) \end{bmatrix} \bullet dt \tag{10}$$

$$\begin{bmatrix} \delta\phi_{bias}(k+1) \\ \delta\theta_{bias}(k+1) \\ \delta\psi_{bias}(k+1) \end{bmatrix} = \begin{bmatrix} \delta\phi_{bias}(k) \\ \delta\theta_{bias}(k) \\ \delta\psi_{bias}(k) \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} \delta V_{xbias}(k+1) \\ \delta V_{ybias}(k+1) \\ \delta V_{zbias}(k+1) \end{bmatrix} = \begin{bmatrix} \delta V_{xbias}(k) \\ \delta V_{ybias}(k) \\ \delta V_{zbias}(k) \end{bmatrix} \tag{12}$$

The general time update equation is

$$P_k = F_{k-1} P_{k-1} F^T{}_{k-1} + G_{k-1} Q_{k-1} G^T{}_{k-1} + Q_s \tag{13}$$

where, P_k is the covariance matrix, $F_k = \left(\frac{\partial f}{\partial x}\right)_k$ is state Jacobian, $G_k = \left(\frac{\partial f}{\partial u}\right)_k$ is control input Jacobian, Q_{k-1} is sensor process noise, Q_s is an additional process noise. The state transform matrix have

$$F_k = \begin{bmatrix} 1 & F_{0,1} & F_{0,2} & F_{0,3} & 0 & 0 & 0 & 0 & 0 & 0 & F_{0,10} & F_{0,11} & F_{0,12} & 0 & 0 & 0 \\ F_{1,0} & 1 & F_{1,2} & F_{1,3} & 0 & 0 & 0 & 0 & 0 & 0 & F_{1,10} & F_{1,11} & F_{1,12} & 0 & 0 & 0 \\ F_{2,0} & F_{2,1} & 1 & F_{2,3} & 0 & 0 & 0 & 0 & 0 & 0 & F_{2,10} & F_{2,11} & F_{2,12} & 0 & 0 & 0 \\ F_{3,0} & F_{3,1} & F_{3,2} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & F_{3,10} & F_{3,11} & F_{3,12} & 0 & 0 & 0 \\ F_{4,0} & F_{4,1} & F_{4,2} & F_{4,3} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & F_{4,13} & F_{4,14} & F_{4,15} \\ F_{5,0} & F_{5,1} & F_{5,2} & F_{5,3} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & F_{5,13} & F_{5,14} & F_{5,15} \\ F_{6,0} & F_{6,1} & F_{6,2} & F_{6,3} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & F_{6,13} & F_{6,14} & F_{6,15} \\ 0 & 0 & 0 & 0 & dt & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & dt & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & dt & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (14)$$

Therefore, we can acquire

$$F_{0,1} = \frac{\partial q_0(t+\Delta t)}{\partial q_1} = \delta\phi_{\ bias}/2 - \delta\phi/2; \quad (15)$$

$$F_{0,2} = \frac{\partial q_0(t+\Delta t)}{\partial q_2} = \delta\theta_{bias}/2 - \delta\theta/2; \quad (16)$$

$$F_{0,3} = \frac{\partial q_0(t+\Delta t)}{\partial q_3} = \delta\psi_{bias}/2 - \delta\psi/2; \quad (17)$$

$$F_{0,10} = \frac{\partial q_0(t+\Delta t)}{\partial \delta\phi_{\ bias}} = \frac{q_1}{2}; \quad (18)$$

$$F_{0,11} = \frac{\partial q_0(t+\Delta t)}{\partial \delta\theta_{bias}} = \frac{q_2}{2}; \quad (19)$$

$$F_{0,12} = \frac{\partial q_0(t+\Delta t)}{\partial \delta\psi_{bias}} = \frac{q_3}{2}; \quad (20)$$

$$F_{1,0} = \frac{\partial q_1(t+\Delta t)}{\partial q_0} = \delta\phi/2 - \delta\phi_{\ bias}/2; \quad (21)$$

$$F_{1,2} = \frac{\partial q_1(t+\Delta t)}{\partial q_2} = \delta\psi/2 - \delta\psi_{bias}/2; \quad (22)$$

$$F_{1,3} = \frac{\partial q_1(t+\Delta t)}{\partial q_3} = \delta\theta_{bias}/2 - \delta\theta/2; \quad (23)$$

$$F_{1,10} = \frac{\partial q_1(t+\Delta t)}{\partial \delta\phi_{\ bias}} = -\frac{q_0}{2} \quad (24)$$

$$F_{1,11} = \frac{\partial q_1(t+\Delta t)}{\partial \delta\theta_{bias}} = \frac{q_3}{2}; \quad (25)$$

$$F_{1,12} = \frac{\partial q_1(t+\Delta t)}{\partial \delta\psi_{bias}} = -\frac{q_2}{2}; \quad (26)$$

$$F_{2,0} = \frac{\partial q_2(t+\Delta t)}{\partial q_0} = \delta\theta/2 - \delta\theta_{\ bias}/2; \tag{27}$$

$$F_{2,1} = \frac{\partial q_2(t+\Delta t)}{\partial q_1} = \delta\psi_{bias}/2 - \delta\psi/2; \tag{28}$$

$$F_{2,3} = \frac{\partial q_2(t+\Delta t)}{\partial q_3} = \delta\phi/2 - \delta\phi_{bias}/2; \tag{29}$$

$$F_{2,10} = \frac{\partial q_2(t+\Delta t)}{\partial \delta\phi_{\ bias}} = -\frac{q_3}{2}; \tag{30}$$

$$F_{2,11} = \frac{\partial q_2(t+\Delta t)}{\partial \delta\theta_{bias}} = -\frac{q_0}{2}; \tag{31}$$

$$F_{2,12} = \frac{\partial q_2(t+\Delta t)}{\partial \delta\psi_{bias}} = \frac{q_1}{2}; \tag{32}$$

$$F_{3,0} = \frac{\partial q_3(t+\Delta t)}{\partial q_0} = \delta\psi/2 - \delta\psi_{bias}/2; \tag{33}$$

$$F_{3,1} = \frac{\partial q_3(t+\Delta t)}{\partial q_1} = \delta\theta/2 - \delta\theta_{bias}/2; \tag{34}$$

$$F_{3,2} = \frac{\partial q_3(t+\Delta t)}{\partial q_2} = \delta\phi_{\ bias}/2 - \delta\phi/2; \tag{35}$$

$$F_{3,10} = \frac{\partial q_3(t+\Delta t)}{\partial \delta\phi_{\ bias}} = \frac{q_2}{2}; \tag{36}$$

$$F_{3,11} = \frac{\partial q_3(t+\Delta t)}{\partial \delta\theta_{bias}} = -\frac{q_1}{2}; \tag{37}$$

$$F_{3,12} = \frac{\partial q_3(t+\Delta t)}{\partial \delta\psi_{bias}} = -\frac{q_0}{2}; \tag{38}$$

$$F_{4,0} = \tfrac{\partial v_x(t+\Delta t)}{\partial q_0} = 2*\mathrm{q0}*(\delta V_x - \delta V_{xbias}) - 2*\mathrm{q3}*(\delta V_y - \delta V_{ybias}) + 2*\mathrm{q2}*(\delta V_z - \delta V_{zbias})\Big); \tag{39}$$

$$F_{4,1} = \tfrac{\partial v_x(t+\Delta t)}{\partial q_1} = 2*\mathrm{q1}*(\delta V_x - \delta V_{xbias}) + 2*\mathrm{q2}*(\delta V_y - \delta V_{ybias}) + 2*\mathrm{q3}*(\delta V_z - \delta V_{zbias})\Big); \tag{40}$$

$$F_{4,2} = \tfrac{\partial v_x(t+\Delta t)}{\partial q_2} = 2*\mathrm{q1}*(\delta V_y - \delta V_{ybias}) - 2*\mathrm{q2}*(\delta V_x - \delta V_{xbias}) + 2*\mathrm{q0}*(\delta V_z - \delta V_{zbias}); \tag{41}$$

$$F_{4,3} = \tfrac{\partial v_x(t+\Delta t)}{\partial q_3} = 2*\mathrm{q1}*(\delta V_z - \delta V_{zbias}) - 2*\mathrm{q0}*(\delta V_y - \delta V_{ybias}) - 2*\mathrm{q3}*(\delta V_x - \delta V_{xbias})\Big); \tag{42}$$

$$F_{4,13} = \frac{\partial v_x(t+\Delta t)}{\partial \delta V_{xbias}} = -q0^2 - q1^2 + q2^2 + q3^2 \tag{43}$$

$$F_{4,14} = \frac{\partial v_x(t+\Delta t)}{\partial \delta V_{ybias}} = 2*\mathrm{q0}*\mathrm{q3} - 2*\mathrm{q1}*\mathrm{q2} \tag{44}$$

$$F_{4,15} = \frac{\partial v_x(t+\Delta t)}{\partial \delta V_{zbias}} = -2(q_0*q_2 + q_1*q_3); \tag{45}$$

$$F_{5,0} = \tfrac{\partial v_y(t+\Delta t)}{\partial q_0} = 2*\mathrm{q3}*(\delta V_x - \delta V_{xbias}) + 2*\mathrm{q0}*(\delta V_y - \delta V_{ybias}) - 2*\mathrm{q1}*(\delta V_z - \delta V_{zbias})\Big); \tag{46}$$

$$F_{5,1} = \tfrac{\partial v_y(t+\Delta t)}{\partial q_1} = 2*\mathrm{q2}*(\delta V_x - \delta V_{xbias}) - 2*\mathrm{q1}*(\delta V_y - \delta V_{ybias}) - 2*\mathrm{q0}*(\delta V_z - \delta V_{zbias}); \tag{47}$$

$$F_{5,2} = \tfrac{\partial v_y(t+\Delta t)}{\partial q_2} = 2*\mathrm{q1}*(\delta V_x - \delta V_{xbias}) + 2*\mathrm{q2}*(\delta V_y - \delta V_{ybias}) + 2*\mathrm{q3}*(\delta V_z - \delta V_{zbias}); \tag{48}$$

$$F_{5,3} = \tfrac{\partial v_y(t+\Delta t)}{\partial q_3} = 2*\mathrm{q0}*(\delta V_x - \delta V_{xbias}) - 2*\mathrm{q3}*(\delta V_y - \delta V_{ybias}) + 2*\mathrm{q2}*(\delta V_z - \delta V_{zbias}); \tag{49}$$

$$F_{5,13} = \frac{\partial v_y(t+\Delta t)}{\partial \delta V_{xbias}} = -2*\mathrm{q0}*\mathrm{q3} - 2*\mathrm{q1}*\mathrm{q2} \tag{50}$$

$$F_{5,14} = \frac{\partial v_y(t+\Delta t)}{\partial \delta V_{ybias}} = -q0^2 + q1^2 - q2^2 + q3^2 \tag{51}$$

$$F_{5,15} = \frac{\partial v_y(t+\Delta t)}{\partial \delta V_{zbias}} = 2(q_0 * q_1 - q_2 * q_3); \tag{52}$$

$$F_{6,0} = \frac{\partial v_z(t+\Delta t)}{\partial q_0} = 2*\mathrm{q1}*(\delta V_y - \delta V_{ybias}) - 2*\mathrm{q2}*(\delta V_x - \delta V_{xbias}) + 2*\mathrm{q0}*(\delta V_z - \delta V_{zbias}); \tag{53}$$

$$F_{6,1} = \frac{\partial v_z(t+\Delta t)}{\partial q_1} = 2*\mathrm{q3}*(\delta V_x - \delta V_{xbias}) + 2*\mathrm{q0}*(\delta V_y - \delta V_{ybias}) - 2*\mathrm{q1}*(\delta V_z - \delta V_{zbias}\Big); \tag{54}$$

$$F_{6,2} = \frac{\partial v_z(t+\Delta t)}{\partial q_2} = 2*\mathrm{q3}*(\delta V_y - \delta V_{ybias}) - 2*\mathrm{q0}*(\delta V_x - \delta V_{xbias}) - 2*\mathrm{q2}*(\delta V_z - \delta V_{zbias}\Big); \tag{55}$$

$$F_{6,3} = \frac{\partial v_z(t+\Delta t)}{\partial q_3} = 2*\mathrm{q1}*(\delta V_x - \delta V_{xbias}) + 2*\mathrm{q2}*(\delta V_y - \delta V_{ybias}) + 2*\mathrm{q3}*(\delta V_z - \delta V_{zbias}\Big); \tag{56}$$

$$F_{6,13} = \frac{\partial v_z(t+\Delta t)}{\partial \delta V_{xbias}} = 2*\mathrm{q0}*\mathrm{q2} - 2*\mathrm{q1}*\mathrm{q3} \tag{57}$$

$$F_{6,14} = \frac{\partial v_z(t+\Delta t)}{\partial \delta V_{ybias}} = -2*\mathrm{q0}*\mathrm{q1} - 2*\mathrm{q2}*\mathrm{q3} \tag{58}$$

$$F_{6,15} = \frac{\partial v_z(t+\Delta t)}{\partial \delta V_{zbias}} = -q0^2 + q1^2 + q2^2 - q3^2 \tag{59}$$

The control input Jacobian matrix is

$$G_k = \begin{bmatrix}
-\frac{q1}{2} & -\frac{q2}{2} & -\frac{q3}{2} & 0 & 0 & 0 \\
\mathrm{SG}(1) & -\frac{q3}{2} & \frac{q2}{2} & 0 & 0 & 0 \\
\frac{q3}{2} & \mathrm{SG}(1) & -\frac{q1}{2} & 0 & 0 & 0 \\
-\frac{q2}{2} & \frac{q1}{2} & \mathrm{SG}(1) & 0 & 0 & 0 \\
0 & 0 & 0 & \mathrm{SG}(4) - \mathrm{SG}(3) - \mathrm{SG}(2) + \mathrm{SG}(5) & \mathrm{SG}(8) - 2*\mathrm{q0}*\mathrm{q3} & \mathrm{SG}(7) + 2*\mathrm{q0}*\mathrm{q2} \\
0 & 0 & 0 & \mathrm{SG}(8) + 2*\mathrm{q0}*\mathrm{q3} & \mathrm{SG}(3) - \mathrm{SG}(2) - \mathrm{SG}(4) + \mathrm{SG}(5) & \mathrm{SG}(6) - 2*\mathrm{q0}*\mathrm{q1} \\
0 & 0 & 0 & \mathrm{SG}(7) - 2*\mathrm{q0}*\mathrm{q2} & \mathrm{SG}(6) + 2*\mathrm{q0}*\mathrm{q1} & \mathrm{SG}(2) - \mathrm{SG}(3) - \mathrm{SG}(4) + \mathrm{SG}(5) \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{60}$$

where

$$SG = \begin{bmatrix} \frac{q_0}{2} \\ {q_3}^2 \\ {q_2}^2 \\ {q_1}^2 \\ {q_0}^2 \\ 2*\mathrm{q_2}*\mathrm{q_3} \\ 2*\mathrm{q_1}*\mathrm{q_3} \\ 2*\mathrm{q_1}*\mathrm{q_2} \end{bmatrix} \tag{61}$$

The process noise matrix is

$$Q_k = \begin{bmatrix} q_{0,0} & SQ(9) & SQ(8) & SQ(7) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ SQ(9) & q_{1,1} & SQ(6) & SQ(5) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ SQ(8) & SQ(6) & q_{2,2} & SQ(4) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ SQ(7) & SQ(5) & SQ(4) & q_{3,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & q_{4,4} & \mathrm{SQ}(3) & \mathrm{SQ}(2) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{SQ}(3) & q_{5,5} & \mathrm{SQ}(1) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{SQ}(2) & \mathrm{SQ}(1) & q_{6,6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (62)$$

where,

$$SQ = \begin{bmatrix} \mathrm{dvzCov} * (\mathrm{SG}(6) - 2*\mathrm{q0}*\mathrm{q1}) * (\mathrm{SG}(2) - \mathrm{SG}(3) - \mathrm{SG}(4) + \mathrm{SG}(5)) - \mathrm{dvyCov} * (\mathrm{SG}(6) + 2*\mathrm{q0}*\mathrm{q1}) * (\mathrm{SG}(2) - \mathrm{SG}(3) + \mathrm{SG}(4) - \mathrm{SG}(5)) + \mathrm{dvxCov} * (\mathrm{SG}(7) - 2*\mathrm{q0}*\mathrm{q2}) * (\mathrm{SG}(8) + 2*\mathrm{q0}*\mathrm{q3}) \\ \mathrm{dvzCov} * (\mathrm{SG}(7) + 2*\mathrm{q0}*\mathrm{q2}) * (\mathrm{SG}(2) - \mathrm{SG}(3) - \mathrm{SG}(4) + \mathrm{SG}(5)) - \mathrm{dvxCov} * (\mathrm{SG}(7) - 2*\mathrm{q0}*\mathrm{q2}) * (\mathrm{SG}(2) + \mathrm{SG}(3) - \mathrm{SG}(4) - \mathrm{SG}(5)) + \mathrm{dvyCov} * (\mathrm{SG}(6) + 2*\mathrm{q0}*\mathrm{q1}) * (\mathrm{SG}(8) - 2*\mathrm{q0}*\mathrm{q3}) \\ \mathrm{dvzCov} * (\mathrm{SG}(6) - 2*\mathrm{q0}*\mathrm{q1}) * (\mathrm{SG}(7) + 2*\mathrm{q0}*\mathrm{q2}) - \mathrm{dvyCov} * (\mathrm{SG}(8) - 2*\mathrm{q0}*\mathrm{q3}) * (\mathrm{SG}(2) - \mathrm{SG}(3) + \mathrm{SG}(4) - \mathrm{SG}(5)) - \mathrm{dvxCov} * (\mathrm{SG}(8) + 2*\mathrm{q0}*\mathrm{q3}) * (\mathrm{SG}(2) + \mathrm{SG}(3) - \mathrm{SG}(4) - \mathrm{SG}(5)) \\ (\mathrm{dayCov}*\mathrm{q1}*\mathrm{SG}(1))/2 - (\mathrm{dazCov}*\mathrm{q1}*\mathrm{SG}(1))/2 - (\mathrm{daxCov}*\mathrm{q2}*\mathrm{q3})/4 \\ (\mathrm{dazCov}*\mathrm{q2}*\mathrm{SG}(1))/2 - (\mathrm{daxCov}*\mathrm{q2}*\mathrm{SG}(1))/2 - (\mathrm{dayCov}*\mathrm{q1}*\mathrm{q3})/4 \\ (\mathrm{daxCov}*\mathrm{q3}*\mathrm{SG}(1))/2 - (\mathrm{dayCov}*\mathrm{q3}*\mathrm{SG}(1))/2 - (\mathrm{dazCov}*\mathrm{q1}*\mathrm{q2})/4 \\ (\mathrm{daxCov}*\mathrm{q1}*\mathrm{q2})/4 - (\mathrm{dazCov}*\mathrm{q3}*\mathrm{SG}(1))/2 - (\mathrm{dayCov}*\mathrm{q1}*\mathrm{q2})/4 \\ (\mathrm{dazCov}*\mathrm{q1}*\mathrm{q3})/4 - (\mathrm{daxCov}*\mathrm{q1}*\mathrm{q3})/4 - (\mathrm{dayCov}*\mathrm{q2}*\mathrm{SG}(1))/2 \\ (\mathrm{dayCov}*\mathrm{q2}*\mathrm{q3})/4 - (\mathrm{daxCov}*\mathrm{q1}*\mathrm{SG}(1))/2 - (\mathrm{dazCov}*\mathrm{q2}*\mathrm{q3})/4 \\ SG(1)^2 \\ q1^2 \end{bmatrix} \quad (63)$$

$$\begin{aligned} q_{0,0} = & \quad (dayCov * q2^2)/4 + (dazCov * q3^2)/4 + (daxCov * SQ(11))/4 \\ q_{1,1} = & \quad (dazCov * q2^2)/4 + (dayCov * q3^2)/4 + daxCov * SQ(10) \\ q_{2,2} = & \quad (daxCov * q3^2)/4 + dayCov * SQ(10) + (dazCov * SQ(11))/4 \\ q_{3,3} = & \quad (daxCov * q2^2)/4 + (dayCov * SQ(11))/4 + dazCov * SQ(10) \\ q_{4,4} = & \quad dvyCov * (SG(8) - 2 * q0 * q3)^2 + dvzCov * (SG(7) + 2 * q0 * q2)^2 \\ & \quad +dvxCov * (SG(2) + SG(3) - SG(4) - SG(5))^2 \\ q_{5,5} = & \quad dvxCov * (SG(8) + 2 * q0 * q3)^2 + dvzCov * (SG(6) - 2 * q0 * q1)^2 \\ & \quad +dvyCov * (SG(2) - SG(3) + SG(4) - SG(5))^2 \\ q_{6,6} = & \quad dvxCov * (SG(7) - 2 * q0 * q2)^2 + dvyCov * (SG(6) + 2 * q0 * q1)^2 \\ & \quad +dvzCov * (SG(2) - SG(3) - SG(4) + SG(5))^2 \end{aligned} \quad (64)$$

where, $daxCov, dayCov, dazCov$ are the radius variance, and $dvxCov$, $dvyCov$, $dvzCov$ are the velocities variance.

The Kalman feedback matrix is

$$K_k = P_k^- H_k \left[H_k P_k^- H_k^T + R_k \right]^{-1} \quad (65)$$

where

$$H_k = \left(\frac{\partial z_p}{\partial x} \right)_k \quad (66)$$

The covariance update equation is

$$P_k^+ = [I - K_k H_k] P_k^- \quad (67)$$

Finally, we could get the state update equation

$$x_k^+ = x_k^- + K_k v \quad (68)$$

The global EKF node is designed in the same method, the details are omitted here. The localization result in the practical environment can be found in Figure 10. The body coordinate represents the filtered pose of the robot now. The robot was running a circle in the practical environment. We can see that the localization performs well even sometimes the GNSS receiver is out of the lock and the result is ready for navigation. The localization and path planning could fail sometimes, even if so many sensors have been deployed. In this application, we designed a remote controller module. Users can control the robot remotely with real-time video streaming.

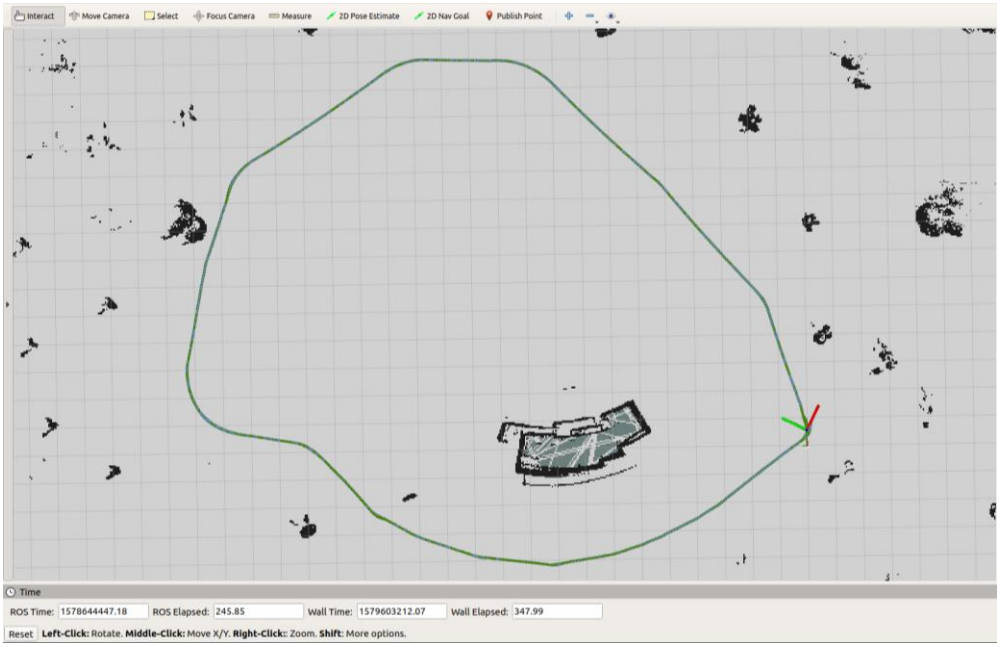

Figure 10. Practical localization performance on map.

The Figure 11 shows a picture of an experiment environment for a surveillance job, this picture is captured from the camera on the robot. In this environment, the road is narrow and the GNSS signal is weak because of many trees. However, the proposed robot system works well. The practical results could be seen in Figure 12. The goal and the blue line are the planned global path. The green line is the local path planning result. We can see the robot locate itself well and find where to go correctly.

Figure 11. Experiment environment.

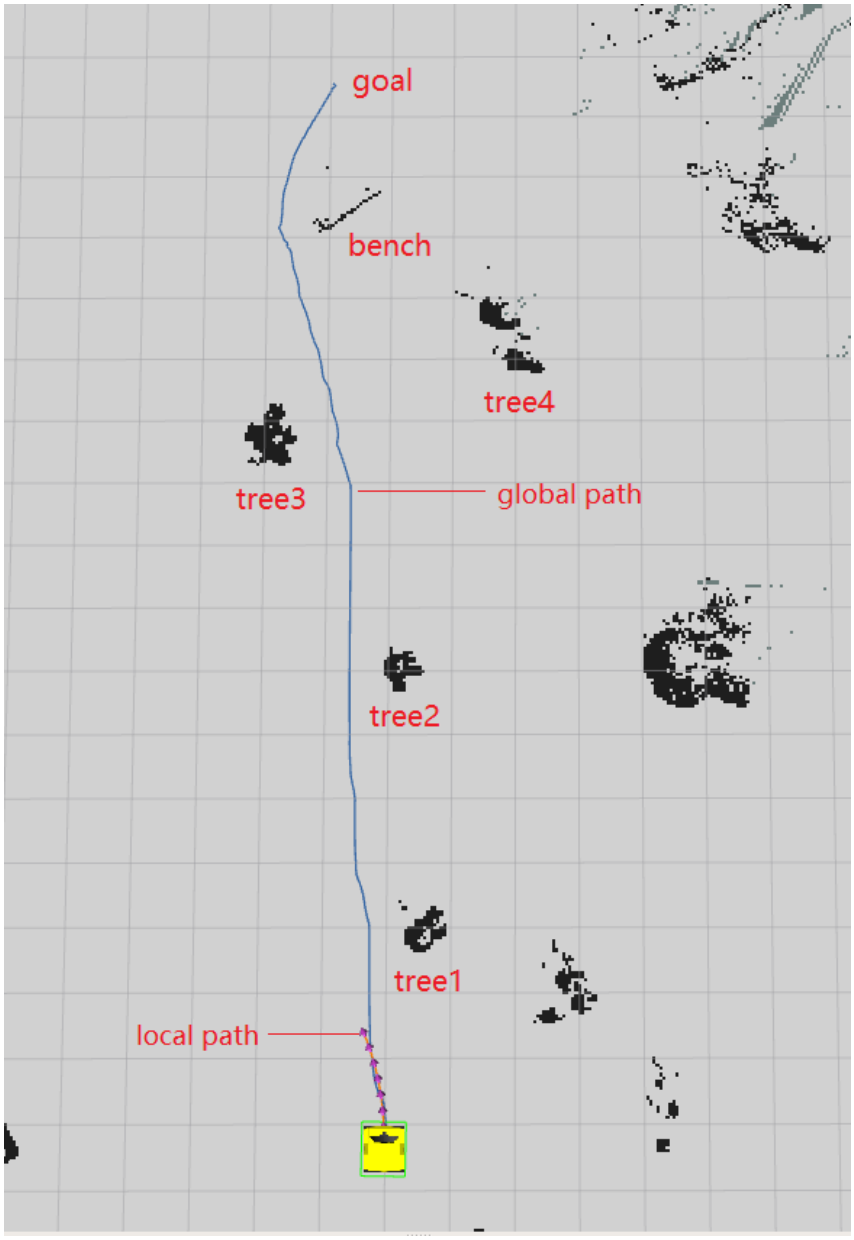

Figure 12. Navigation results.

7. Environment Inspection

The dam crack identification and pedestrian detection are the key technologies related to the safety of the dam. We use the trained model of YOLO V3 [31] to recognize them. We use the dataset online to train the YOLO V3 algorithm and then recognize the dam crack. The YOLO V3 algorithm includes a darknet-53 module, eight DBL components, three convolutional layers, two upsampling layers, and two tensor concat layers. The darknet-53 includes a DBL component and five residual learning units res1, res2, res8, res8, and res4.

The DBL contains a convolutional layer, a BN layer and a leaky ReLU. The convolutional parameters of the convolutional layer are the kernel size 3×3, stride 1, same padding and output channels 32. The res1, res2, res8, and res4 contain 1, 2, 8, and 4 basic units of the residual learning, respectively, and each basic unit of the residual learning contains two DBL and an identity map. The convolutional parameters of the first DBL are the kernel size 1×1, stride 1, and the output channels are equal to the number of basic units of the residual learning. The convolutional parameters of the

second DBL are the kernel size 3 × 3, stride 2, same padding, and output channels are equal to 2 times basic units of the residual learning.

The three convolutional layers are CConv1, CConv2, and CConv3. The filter sizes of the CConv1, CConv2, and CConv3 are 512, 256, and 128 respectively. The convolutional layers CConv1, CConv2, and CConv3 respectively contain six convolutional operations. The convolutional parameters of the first convolutional operation are the convolutional kernel size 1 × 1, and the number of output channels is equal to the filter size. The convolutional parameters of the second convolutional operation are the convolutional kernel size 3 × 3, and the number of output channels is equal to 2 times the filter size. The convolutional parameters of the third convolutional operation are the convolutional kernel size 1 × 1, and the number of output channels is equal to the filter size. The convolutional parameters of the fourth convolutional operation are the convolutional kernel size 3 × 3, and the number of output channels is equal to 2 times the filter size. The convolutional parameters of the fifth convolutional operation are the convolutional kernel size 1 × 1, and the number of output channels is equal to the filter size. The convolutional parameters of the sixth convolutional operation are the convolutional kernel size 3 × 3, and the number of output channels is equal to 2 times the filter size. The upsampling layers Upsample1 and Upsample2 sample the input feature maps and the input feature maps of the concat layer into the same size.

The network structure is shown in Figure 13. This method can do the detection job at 10 Hz with less than 20% CPU occupying. Some results are shown in Figures 14 and 15.

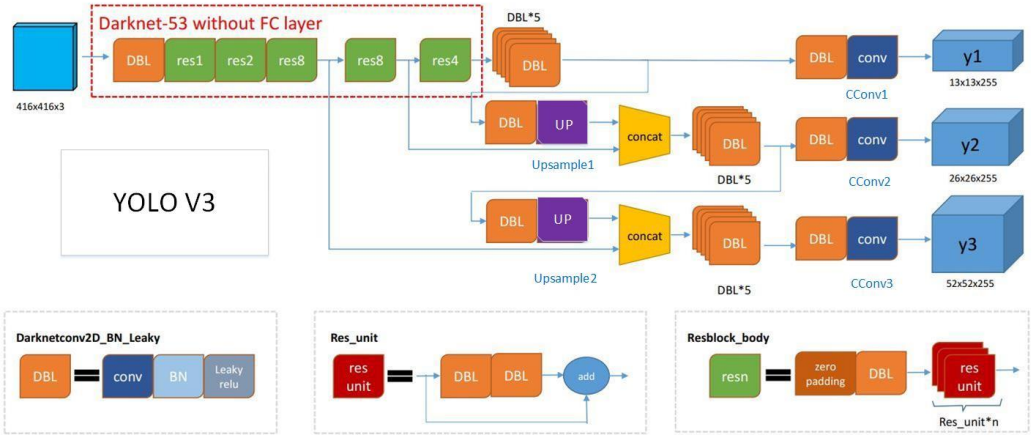

Figure 13. YOLO V3 network structure.

Figure 14. People detection results.

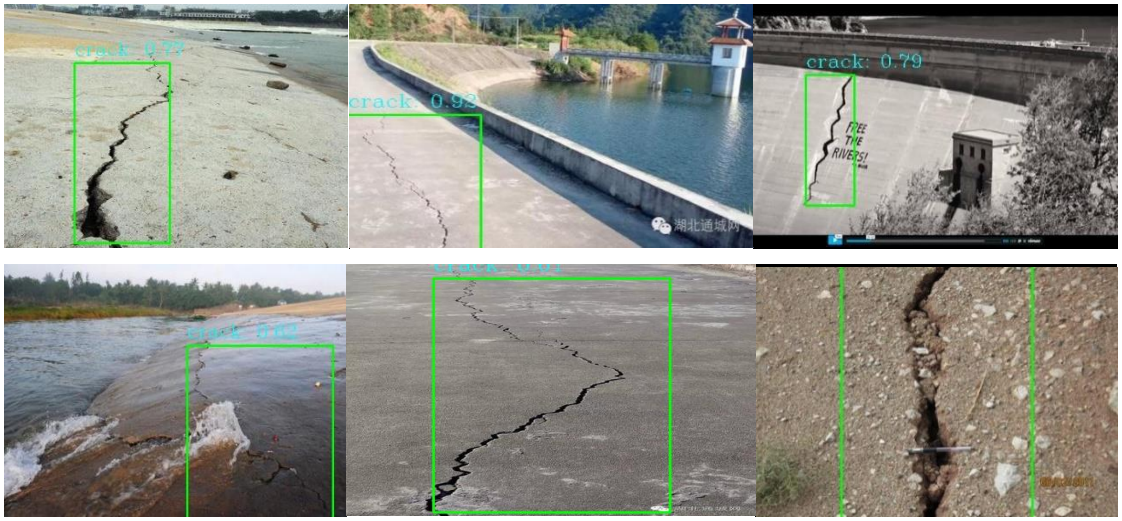

Figure 15. Dam crack detection results.

8. Conclusions

In this paper, a general robot system is proposed for dam surveillance. The robot is connected to cloud servers and terminal users by mobile internet and IoT network. Like the robot itself, this paper introduces mechanics layout, sensor selection, and the navigation method. A simple controller and a wheel odometry calculation are proposed and achieve good performance. A two-node EKF structure localization framework is proposed to solve localization problem, which fuses LiDAR SLAM, wheel odometry, IMU, and GNSS signals. For unexpected circumstances, a remote controller based on real-time video streaming is deployed as an emergency supplement. To make the whole system able to work all-time, a control state machine is also introduced. A YOLO v3 network is trained and deployed to detect dam crack and people around. From the practical experiments, this system can work well and is capable of the surveillance job for dams. Afterward, we will pay more attention to specific dams' surveillance jobs, such as intrusion detection and dam deformation detection. We believe robots will greatly improve the work efficiency for water conservancy in the future.

Author Contributions: Data curation, C.Z. and Q.W.; Formal analysis, T.H. and H.W.; Funding acquisition, Q.Z.; Methodology, C.Z., Y.A., and Q.W.; Software, Y.A. and H.W.; Supervision, Q.Z.; Writing—original draft, C.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Key R&D Program of China under grant no. 2018YFC040770* and the National Natural Science Foundation of China under grant no. 61673083.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Su, H.; Hu, J.; Wu, Z. A study of safety evaluation and early-warning method for dam global behavior. *Struct. Health Monit.* **2012**, *11*, 269–279.
2. Räty, T.D. Survey on contemporary remote surveillance systems for public safety. *IEEE Trans. Syst. Man Cybern. Part C (Appl. and Rev.)* **2010**, *40*, 493–515. [CrossRef]
3. Theodoridis, T.; Hu, H. Toward intelligent security robots: A survey. *IEEE Trans. Syst. Man Cybern. Part C (Appl. and Rev.)* **2012**, *42*, 1219–1230. [CrossRef]
4. Yoshida, T.; Nagatani, K.; Tadokoro, S.; Nishimura, T.; Koyanagi, E. Improvements to the rescue robot quince toward future indoor surveillance missions in the Fukushima Daiichi nuclear power plant. In *Field and Service Robotics*; Kazuya, Y., Satoshi, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 19–32.
5. Debenest, P.; Guarnieri, M. Expliner—From prototype towards a practical robot for inspection of high-voltage lines. In Proceedings of the 2010 IEEE 1st International Conference on Applied Robotics for the Power Industry, Montreal, QC, Canada, 5–7 October 2010; pp. 1–6.

6. Hess, W.; Kohler, D.; Rapp, H.; Andor, D. Real-time loop closure in 2D LIDAR SLAM. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1271–1278.
7. Grisetti, G.; Stachniss, C.; Burgard, W. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans. Rob.* **2007**, *23*, 34–36. [CrossRef]
8. Qin, T.; Li, P.; Shen, S. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Rob.* **2018**, *34*, 1004–1020. [CrossRef]
9. Leutenegger, S.; Lynen, S.; Bosse, M.; Siegwart, R.; Furgale, P. Keyframe-based visual-inertial odometry using nonlinear optimization. *Int. J. Rob. Res.* **2015**, *34*, 314–334. [CrossRef]
10. Kunchev, V.; Jain, L.; Ivancevic, V.; Finn, A. Path planning and obstacle avoidance for autonomous mobile robots: A review. In *Knowledge-Based Intelligent Information and Engineering Systems*; Gabrys, B., Howlett, R.J., Jain, L.C., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 537–544.
11. Duchoň, F.; Babinec, A.; Kajan, M.; Beňo, P.; Florek, M.; Fico, T.; Jurišica, L. Path planning with modified a star algorithm for a mobile robot. *Procedia Eng.* **2014**, *96*, 59–69. [CrossRef]
12. Jiang, C.G.; Peng, J.G. Research, manufacture and application of GPS-based surveying robot automatic monitoring system for dam safety. In Proceedings of the 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, Shanghai, China, 20–22 November 2009; pp. 151–155.
13. Kazmi, W.; Ridao, P.; Ribas, D.; Hernández, E. Dam wall detection and tracking using a mechanically scanned imaging sonar. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 3595–3600.
14. Buffi, G.; Manciola, P.; Grassi, S.; Barberini, M.; Gambi, A. Survey of the Ridracoli Dam: UAV–based photogrammetry and traditional topographic techniques in the inspection of vertical structures. *Geomatics Nat. Hazard. Risk* **2017**, *8*, 1562–1579. [CrossRef]
15. Ridolfi, E.; Buffi, G.; Venturi, S.; Manciola, P. Accuracy analysis of a dam model from drone surveys. *Sensors* **2017**, *17*, 1777. [CrossRef]
16. Yi, J.; Wang, H.; Zhang, J.; Song, D.; Jayasuriya, S.; Liu, J. Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation. *IEEE Trans. Rob.* **2009**, *25*, 1087–1097.
17. Mandow, A.; Martinez, J.L.; Morales, J.; Blanco, J.L.; Garcia-Cerezo, A.; Gonzalez, J. Experimental kinematics for wheeled skid-steer mobile robots. In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2007; pp. 1222–1227.
18. Kozłowski, K.; Pazderski, D. Modeling and control of a 4-wheel skid-steering mobile robot. *Int. J. Appl. Math. Comput. Sci.* **2004**, *14*, 477–496.
19. Wang, T.; Wu, Y.; Liang, J.; Han, C.; Chen, J.; Zhao, Q. Analysis and experimental kinematics of a skid-steering wheeled robot based on a laser scanner sensor. *Sensors* **2015**, *15*, 9681–9702. [CrossRef] [PubMed]
20. Lenain, R.; Thuilot, B.; Cariou, C.; Martinet, P. Rejection of sliding effects in car like robot control: Application to farm vehicle guidance using a single RTK GPS sensor. In Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), Las Vegas, NV, USA, 27–31 October 2003; pp. 3811–3816.
21. Zhang, J.; Singh, S. LOAM: Lidar Odometry and Mapping in Real-time. *Rob. Sci. Syst.* **2014**, *2*, 9.
22. Lin, J.; Zhang, F. A fast, complete, point cloud based loop closure for LiDAR odometry and mapping. *arXiv* **2019**, arXiv:1909.11811.
23. Sabet, M.T.; Daniali, H.M.; Fathi, A.R.; Alizadeh, E. Experimental analysis of a low-cost dead reckoning navigation system for a land vehicle using a robust AHRS. *Rob. Autom. Syst.* **2017**, *95*, 37–51. [CrossRef]
24. Surber, J.; Teixeira, L.; Chli, M. Robust visual-inertial localization with weak GPS priors for repetitive UAV flights. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 6300–6306.
25. Alejo, D.; Caballero, F.; Merino, L. A Robust Localization System for Inspection Robots in Sewer Networks. *Sensors* **2019**, *19*, 4946. [CrossRef] [PubMed]
26. Min, H.; Wu, X.; Cheng, C.; Zhao, X. Kinematic and Dynamic Vehicle Model-Assisted Global Positioning Method for Autonomous Vehicles with Low-Cost GPS/Camera/In-Vehicle Sensors. *Sensors* **2019**, *19*, 5430. [CrossRef]

27. Rezaei, S.; Sengupta, R. Kalman filter-based integration of DGPS and vehicle sensors for localization. *IEEE Trans. Control Syst. Technol.* **2007**, *15*, 1080–1088. [CrossRef]
28. Suhr, J.K.; Jang, J.; Min, D.; Jung, H.G. Sensor fusion-based low-cost vehicle localization system for complex urban environments. *IEEE Trans. Intell. Transp. Syst.* **2016**, *18*, 1078–1086. [CrossRef]
29. Treptow, A.; Cielniak, G.; Duckett, T. Active people recognition using thermal and grey images on a mobile security robot. In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada, 2–6 August 2005; pp. 2103–2108.
30. Cha, Y.J.; Choi, W.; Büyüköztürk, O. Deep learning-based crack damage detection using convolutional neural networks. *Comput.-Aided Civ. Infrastruct. Eng.* **2017**, *32*, 361–378. [CrossRef]
31. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.