SoC Implementation of Visual-inertial Odometry for Low-cost Ground Robots

Bo Liu¹, Lin Li² and Hengzhu Liu*¹

¹ School of Computer, National University of Defense Technology, Changsha, Hunan, 410073, China
² School of Computer, National University of Defense Technology, Changsha, Hunan, 410073, China
*Corresponding author’s e-mail: liubo17d@nudt.edu.cn

Abstract. Simultaneous localization and mapping (SLAM) is a key technique for autonomous navigation of a robot in unknown environments. Recently, many researchers focus on the algorithm optimization of visual-inertial odometry (VIO) as a real-time SLAM method. In this paper, different from pure algorithm research, we propose a novel low-cost hardware solution based on the system on a chip (SoC) to achieve real-time performance and low power consumption of VIO. We design a novel hardware architecture to accelerate a popular VIO algorithm, with the time-costly image feature processing computed on FPGA and floating-point numerical computation been done on ARM. On the algorithm side, we fuse odometry information from the two wheels of the ground robot in a unified optimization method, which results in accurate metric scale estimation of the VIO. The experimental results show that our design and implementation have superior performance for ground robot applications, allowing real-time localization of a robot with a very low-cost hardware for computation in complex environments.

1. Introduction
Simulation localization and mapping (SLAM) are the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent’s location within it [1-4]. For robots, SLAM plays an important role in some basic problems, especially in localization, tracking, and path planning. Recently, using the camera as an external sensor becomes popular for its rich visual information and low hardware cost. And there are plenty of researches devoted to the study about visual simultaneous localization and mapping (VSLAM) and they propose some effective algorithm, such as ORB-SLAM [5], RTAB-Map [6], VINS-mono [7] and so on.

Due to the problem of real-time and power consumption, many algorithms of VSLAM are difficult to be implemented on low-cost robots and many researches put their attention on the hardware accelerator. Related to SLAM algorithms, the operation of feature extraction is the bottleneck of performance and energy consumption. So, FPGAs have been used to implement feature detectors based on ORB [8], SURF [9], SIFT [10] and so on. [11] implements FPGA-based ORB feature extraction at 20 fps at a resolution of 640 x 480, which strikes a great balance between performance and energy consumption. [12] also designs an FPGA acceleration of multilevel ORB feature extraction with high performance at 72 fps at a resolution of 1920 x 1080. Besides, there are some solutions achieving the whole SLAM system on hardware. Lee et al. [13] demonstrates a lightweight approach to low-cost
indoor robot localization, based on line features and a simplistic 2D map. And their robot can run at 2.5 fps on an ARM processor at 533MHz, while currently building a map and planning a path which can avoid obstacle. [15] presents an implementation of LSD-SLAM [14] on an FPGA System on Chip, running at more than 4 fps at a resolution of 320 × 240.

VINS-mono [7] proposes a robust and versatile monocular visual-inertial estimator, providing state-of-the-art solutions to IMU pre-integration, estimator initialization and failure recovery, online extrinsic calibration, tightly-coupled visual-inertial odometry, re-localization, and global optimization. So, we implement our system based on VINS-mono. We summarize our contributions as follows:

• We propose a new method to combine the VIO algorithm with odometry of a ground robot.
• We propose a novel hardware architecture for our algorithm based on FPGA SoC to achieve real-time performance on very low-cost hardware. The implementation is capable of running feature tracking at more than 20 fps at a solution 640×480 with localization and mapping at more than 7 fps.

2. Overview of VINS-mono

VINS-mono has been successfully applied to small-scale AR scenarios, medium-scale drone navigation, and large-scale state estimation tasks. But for some low-cost mobile robots, it may not be suitable. So, we changed the partial structure of the algorithm to make it better on low-cost mobile robots. The input includes camera(30hz), IMU (100hz) and odometry(100hz), and the output is the robot pose. The structure of our modified monocular visual-inertial state estimator is shown in Fig.1.

As shown in the Fig.1, our system is based on the VINS-mono. But we fused the odometry information in our system and our method is much suitable for the low-cost ground robots. In our system, there are mainly three modules: measurement pre-processing, estimator initialization and tightly-coupled monocular VIO (Visual Inertial Odometry).

• The system begins with measurement pre-processing. In this module, feature points are extracted, tracked and error points are eliminated by RANSAC, the IMU measurements and odometry measurements between two consecutive frames are pre-integrated and recorded respectively. The input of this module is the images captured by camera and the IMU and odometry measurements, and the output is the coordinates of feature points and the pre-integrated result.
• Estimator initialization modules generate all necessary values for bootstrapping the subsequent nonlinear optimization-based VIO, such as pose, velocity, gravity vector, gyroscope bias, and 3D feature location. This module is the same as the VINS-mono.
• The VIO module tightly fuses pre-integrated IMU measurements, odometry measurements and feature observations. The input of this module includes the feature points, IMU and odometry information and so on, the output is the robot pose.

In our system, the data from the IMU measurements and the odometry measurements are unreliable, so we fuse the information with visual information. The VIO module fuses IMU measurements, feature observations and odometry measurements, the algorithm is introduced in section 3.1. And we proposed a novel hardware architecture for the measurement pre-processing module, which is introduced in
section 3.3. Due to the complexity and calculation of re-localization and global optimization modules, we do not implement other modules in our system, including the pose graph optimization module and the re-localization module. The total architecture is introduced in section 3.2.

3. Method

3.1. Odometry information

Due to the purely projective nature of a single camera, motion estimates and map structure can only be recovered up to scale. Scale issue is a classic problem in monocular camera. Although VINS-mono [7] designs the re-localization and graph optimization modules to eliminate drift, scale drift still exists. And the scale error will have a serious impact on the results when the noise greatly interferes with the IMU measurement.

Odometry is used in robotics by some legged or wheeled robots to estimate their localization relative to a starting location. We combine odometry with VINS to continuously modify the scale factor.

Like VINS-mono, we also use the tightly coupled monocular VIO based on sliding window for high precision and robust state estimation. We consider $\dot{\rho}^w$ as the world frame and $\dot{\rho}^b$ as the body frame, $q_b^w, p_b^w$ are rotation and translation from the body frame to the world frame. Then the full state vector in the sliding window is defined as:

$$\chi = [x_0, x_1, \ldots, x_n, \lambda_0, \lambda_1, \ldots, \lambda_m]$$

$$x_k = [p_{bk}, v_{bk}, q_{bk}, b_a, b_g]$$

$$x_b^0 = [p_b^0, q_b^0]$$

where $\chi$ is the set of state vector, $x_k$ is the IMU state at the time that the kth image is captured and $\lambda_i$ is the inverse depth of the lth feature from its first observation. $b_k$ is the body frame while taking the kth image. The IMU state includes five parameters: localization $p_{bk}^w$, velocity $v_{bk}^w$, orientation $q_{bk}^w$ of the IMU in the world frame, acceleration bias $b_a$ and gyro scope bias $b_g$ in the IMU body frame. $n$ is the total number of keyframes, and $m$ is the total number of features in the sliding window.

if we suppose $\dot{\rho}$ as the estimate of a certain quantity and minimize the sum of prior and the Mahalanobis norm of all measurement residuals, we can obtain a maximum posteriori estimation:

$$\min_{\chi} \{||r_p - H_{\rho} \chi||^2 + \sum_{k \in \beta} ||r_{\beta}(2_{bk}^*, \chi)||^2_{p_{bk}} + \sum_{(l,j) \in C} ||r_C(\hat{z}_{ci}^l, \chi)||^2_{p_l} + \sum_{i \in D} ||r_D(\hat{z}_{ci}^{j+1}, \chi)||^2\}$$

Where the Huber norm [18] is defined as:

$$\rho(\delta) = \begin{cases} \frac{\delta^2}{2s} & s < 1 \\ \delta & s \geq 1 \end{cases}$$

$r_\beta(2_{bk}^*, \chi), r_C(\hat{z}_{ci}^l, \chi)$ and $r_D(\hat{z}_{ci}^{j+1}, \chi)$ are residuals for IMU, visual and odometry measurements respectively. $\beta$ is the set of all IMU measurements, $C$ is the set of features which have been observed at least twice in the current sliding window and $D$ is the set of odometry measurements between two adjacent key frames. $r_p, H_{\rho}$ is the prior information from marginalization.

The residuals for IMU and visual measurements are the same as VINS-mono [7], So we won’t go into too much details, and let’s focus on the calculation of variance. Consider the odometry measurements within two consecutive frames $b_k$ and $b_{k+1}$. In the sliding window, the residuals for odometry measurements can be define as following:

$$r_D(\hat{z}_{ci}^{j+1}, \chi) = \left||p_{bi}^w - p_{bi+1}^w\right|^2 - d_{odom}^2 = (u_i - u_{i+1})^2 + (v_i - v_{i+1})^2 + (w_i - w_{i+1})^2 - d_{odom}^2$$

Where $p_{bi}^w$ is the localization element of the i-th frame state. And $u, v, w$ are the three axes of the rectangular coordinate system in space. We can use non-linear least squares to get the solution of this nonlinear problem.
3.2. the implementation of hardware accelerator

At first, we run VINS-mono on the Odroid XU4 by disabling loop closure, and the image frames are processed at 8 fps on average. However, the performance is not so good and the result is unstable with big drift.

After delving into the problem, we found that feature extraction and tracking are indeed the bottleneck of performance and energy consumption. For each new image, the KLT sparse optical flow algorithm [16] tracks the existing features, new features [17] are detected to ensure the minimum number (100-300) of each image feature.

The KLT tracker uses three assumptions to make feature tracking a solvable problem: brightness consistency, spatial coherence and temporal persistence. If the latency of features tracking is too long, the premises are no longer true. So, the frame rate is very important for feature detection and tracking and this part of the operation needs to be as fast as possible to ensure that there are enough common features between the two consecutive key frames.

When the image processing rate is higher than 10Hz, the result is normal and stable. However, it is difficult to meet this requirement on low-cost embedded platforms, such as Odroid XU4. So, we decide to design a novel hardware architecture based on SoC to accelerate the calculation.

![Image of the total structure chart of the proposed hardware accelerator.]

The structure of our design based on SoC is shown as Fig.2. We only implement the measurement preprocessing module on FPGA to accelerate the operation about feature detection and tracking and ensure the reliability of the three assumptions. The input of FPGA is the image which is captured by the camera and the output is the coordinates of feature points which will be used for the calculation of visual odometry measurements. For the estimator initialization module, we deploy it on the ARM because it doesn’t need to be run many times and the latency of this modules will not have a bad influence on the result. Besides, the monocular VIO module is also worked on the ARM processor, and the frame rate is enough to get the high accuracy. The input of ARM is the coordinates of feature points and IMU and odometry measurements, and the output is the robot pose.

In our system, there are two input interfaces, visual sensor input IMU and odometry measurement input. After the image is captured by visual sensor, visual data are transmitted to the programmable logic (PL) for feature detection and tracking, which mainly contains four parts, feature detection, image pyramid, feature tracking and feature selection.

- The input of feature detection is the visual sensor data, the output of feature point is conserved in the DDR.
- The input of image pyramid is the same as the part of feature detection and four layers of image is outputted to the memory.
- The part of feature tracking reads the image layer in sequence and export the feature point tracked by the KLT sparse optical flow algorithm [19].
- The part of feature selection filtrates the new feature point generated by the part of feature detection and ensures the minimum number of each image feature.
And after the IMU and odometry data are transmitted to the PL which is mainly implemented by FPGA, PL transform and record that information in the memory. The operations of image consume too many computer resources and the programmable logics cannot deal with the operation at once, so we save the intermediate result in the DDR. While detecting the signal that the FPGA has finished its work, the processing system (PS) begins to receive the data about the coordinates of feature points, IMU and odometry information from FPGA continuously by reading the data in the DDR and AXI-GD port. And the PS mainly includes the software part which is implemented by high-level language on ARM, such as C++. In this part, the estimator initialization module and the monocular VIO module are deployed, all of the information will be fused and the system will output the pose of robot.

3.3. hardware design for feature tracking

For the feature tracking, we propose a novel hardware architecture to accelerate the operation. The algorithm is the Lucas–Kanade method [20], which is a widely used differential method for optical flow estimation developed by Bruce D. Lucas and Takeo Kanade. The input of this parts is two frames of pictures, which includes several layers of image pyramids. We suppose that the feature points \((x_0, y_0)\) of the current frame are known, and we need to find the corresponding points \((x_1, y_1)\) in the next frame.

- Starting from the top-level pyramid image, the \(64 \times 64\) pixel size matrix near the current feature point in the top-level pyramid of the current frame is first pulled out, and then the algorithm calls the \(64 \times 64\) pixel size matrix near the current feature point in the top-level pyramid of the next frame.
- We assume that the position of the feature points in the adjacent frames is the same, that is, \(x_0 = x_1, y_0 = y_1\). Then we compare the \(21 \times 21\) pixel size matrix near the two feature points to see the difference. If the difference exceeds the threshold, we think that difference is relatively large. \((x_1, y_1)\) is considered to have moved relative to \((x_0, y_0)\). According to the difference, we can obtain the offset between two point, \((\delta y, \delta y)\). Then let \(x_1 = x_1 + \delta x, y_1 = y_1 + \delta y\), get a new \((x_1, y_1)\), repeat this step for iteration.
- After the comparison, difference is less than the threshold value, it is considered that \((x_0, y_0)\) is very similar to the latest \((x_1, y_1)\), and \((x_1, y_1)\) can be output to the next layer of pyramid data for the next iteration loop. If the number of iterations reaches the specified number and the comparison difference is still greater than the threshold, the feature point is considered to be lost.
- By traversing each feature point, and the corresponding positions of all feature points on the next image frame can be determined.

![Fig. 3. the layout design for the feature tracking](image)

the structure of hardware accelerate is shown as Fig.3. Because a boost is set to 64 pixels on the FPGA, the operation of \(64 \times 64\) data block is adopted, which greatly reduces the bandwidth resource. The whole design resource consumption is very small, the speed can reach 150M clock. Besides, the
accuracy of feature point tracking is very high, and even if it is put on the fisheye lens with large distortion, high quality feature point results can be obtained.

4. Experimental Result
This visual inertial odometry is implemented and evaluated on the Xilinx Zynq-7020 SoC board, which is a low-cost development board. The feature detection and track of images are deployed on FPGA to obtain the low latency, and other parts is implemented on ARM. So, they have different latency and throughput. The computer resource consumed by the proposed hardware is shown in table.1.

|             | Total number | We need          | Total number | We need          |
|-------------|--------------|------------------|--------------|------------------|
| DSP         | 220          | 69(31.36%)       | 106400       | 40965(38.5%)     |
| BRAM (36kb) | 140          | 36(45.00%)       | 32           | 9(28.13%)        |
| LUTs        | 53200        | 28767(54.07%)    | 9            | 2.742W           |

We process image captured by camera at a resolution of 640 × 680. The clock frequency of the hardware is 150MHz, and the proposed hardware can detect and track feature at 20 fps, while the frame rate of other parts is at 7 fps. And the frame rate of our method is sufficient for high precision localization and mapping. This result indicates that our method is a great balance between performance and computer resource consumption.

For each frame, the running time is not only related to our pre-set parameters, but also to the number of feature points detected from the image. We supposed that the number of feature points is 200 and the level of image pyramid is 3, then the operations on FPGA needs 0.0502s for each frame and the operations on ARM needs 0.135s.

![Fig. 4. the comparison of experimental results between ours and the Odroid XU4. The right is ours and the left is the Odroid XU4 board.](image)

We test our method in the indoor environment and we choose our laboratory environment as the experiment area which is shown in Fig.4. We control the mobile robot to move back and forth, from the starting line, the robot move forward \( n \) meters, transversely offset \( m \) meters, and then turn around and move forward \( n \) meters back to the starting line. We repeat this process several times and record the trajectory of the robot which is shown in Fig.4.

We use the serial port for debugging, so the data are not recorded until the robot moves for a while, so the localization of the robot at the beginning is not recorded. As can be seen from the trajectory map, there is almost no deviation in the localization result of the robot and our method is accurate and enough to map and locate at a real-time level.

5. Conclusion
We solve the problem about how the mobile robot map and localization on embedded platform and we come up with a low-cost solution. We propose a novel hardware accelerator for the VINS-mono with odometry that is used for mobile robot to map and localization. As far as I know, this is the first time someone has implemented the VINS-mono algorithm in a hardware way. The evaluation results have shown that the proposed accelerator could achieve feature detection and tracking at a resolution of
640 × 480 at 20hz and VIO at 7hz, which is enough to map and locate at a real-time level. And this accelerator can be used for various kinds of mobile robots, such as automated guided vehicle (AGV).

References
[1] Durrant-Whyte H , Bailey T.(2006) Simultaneous localization and mapping Part I. IEEE J. IEEE Robotics & Automation Magazine, (DOI:10.1109/ISSN.1070-9932.)
[2] Bailey T, Durrant-Whyte H. (2006) Simultaneous localization and mapping (SLAM) part II. IEEE J. IEEE Robotics & Automation Magazine, (DOI:10.1109/ISSN.1070-9932.)
[3] Cadena C , Carlone L, Carrillo Het al. (2016) Past, Present, and Future of Simultaneous Localization and Mapping Toward the Robust-Perception Age IEEE J. IEEE Transactions on Robotics, (DOI:10.1109/ISSN.1552-3098.)
[4] Perera S , Barnes D N , Zelinsky D A. (2014) Exploration: Simultaneous Localization and Mapping (SLAM) Computer Vision: A Reference Guide, Springer US, pp., (DOI:10.1007/ISBN.9780387314396)
[5] Mur-Artal, Raúl, J. M. M. Montiel, and J. D. Tardós. (2017) ORB-SLAM: A Versatile and Accurate Monocular SLAM System. IEEE Transactions on Robotics
[6] Das S. (2018) Simultaneous Localization and Mapping (SLAM) using RTAB-MAP[J].
[7] Tong Q, Li P, Shen S. (2017) VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. IEEE J. IEEE Transactions on Robotics.
[8] Rublee E , Rabaud V , Konolige Ket al. (2012) ORB: An efficient alternative to SIFT or SURF. In: International Conference on Computer Vision.
[9] Bay, Herbert et al. (2008) Speeded-Up Robust Features (SURF). Computer Vision & Image Understanding.
[10] Lowe D G. (2004) Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision.
[11] Fang, Weikang et al (2017) FPGA-based ORB Feature Extraction for Real-Time Visual SLAM.
[12] Weberruss, Josh et al.(2017) FPGA acceleration of multilevel ORB feature extraction for computer vision". International Conference on Field Programmable Logic & Applications IEEE.
[13] Lee, Seongsoo , S. Lee , and J. J. Yoon. (2012) Illumination-Invariant Localization Based on Upward Looking Scenes for Low-Cost Indoor Robots. Advanced Robotics.
[14] Engel, Jakob , Thomas Schöps, and D. Cremers. (2014) LSD-SLAM: Large-Scale Direct Monocular SLAM. In:European Conference on Computer Vision Springer.
[15] Boikos, Konstantinos, and C. S. Bouganis. (2016) Semi-dense SLAM on an FPGA SoC. In: International Conference on Field Programmable Logic & Applications .
[16] Stan Birchfield. (2007) KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker.
[17] Chen, Zheng Guo et al. (2014) Using FPGA to Accelerate Deduplication on High-Performance SSD. Advanced Materials Research 1042:212-217.
[18] P. Huber. (1964) Robust estimation of a location parameter. Annals of Mathematical Statistics.
[19] Li, Peiliang et al. (2017) Monocular Visual-Inertial State Estimation for Mobile Augmented Reality’. 2017 IEEE International Symposium on Mixed and Augmented Reality {DOI:10.1109/ISMAR.2017.18}. 