ATLANTIS PRESS

Advances in Engineering Research, volume 149

2nd International Conference on Mechanical, Electronic, Control and Automation Engineering (MECAE 2018)

Design of Indoor Real-time Positioning on Embedded Platform

Xin Shu[1, 2], Chang Liu[1], Haoyuan Cai[1, a] and Tianyang Cao[1]

[1]State Key Laboratory of Transducer, Institute of Electronics, Chinese Academy of Sciences, Beijing 100190, China;

[2] University of Chinese Academy of Sciences, Beijing 100049, China.

[a]hycai@mail.ie.ac.cn

Keywords: Indoor Positioning, Embedded Platform, Inertial, Vision.

Abstract. Indoor positioning is becoming a very popular research direction in recent years. In this paper, we realize real-time trajectory calculation through the fusion of inertial and vision. We have two experiments in different environments and we find that the algorithm's error is below 50% of pure inertial at least. It means that the algorithm mentioned in this paper has a higher accuracy. Besides, we run this algorithm on an embedded platform built by ourselves. This platform is based on raspberry pi, which greatly reduces costs of indoor positioning. In this way, algorithm need not to be ran on the bulky computer.

1. Introduction

With the rapid development of science and technology, automation is playing a more and more important role in lots of fields. In many cases, we want to use indoor positioning technology to get the object's location information. Vision is an emerging solution of robot indoor positioning technology, which can build a map according to the environment and realize the function of autonomous positioning. It can be divided into dense method, sparse method and semi-dense method [1]. Dense method considers all pixels in image as the research object, which can improve the utilization of image [2]. Sparse method only extracts the feature points of image. And semi-dense method use the boundary information of image to improve processing speed [3]. At the same time, traditional personal computer is not a great image processing platform because of its inconvenient. In this paper, we use raspberry pi as the image processing embedded platform, intended to design a real-time indoor positioning solution. Compared to other ways, it has the advantages of cheap and convenient. By the way, it can get the location information in real-time.

2. Building of Embedded Platform

Since the appearance of raspberry pi in 2012, it has been sought-after by computer enthusiasts. It includes an ARM as the central processing unit and a SD card as the memory. It also includes several USB ports and a 10/100 Ethernet port, so that it can work with the mouse, the keyboard, the wireless LAN and the Ethernet cable. By the way, it has a video analog signal output of TV and High Definition Multimedia Interface, to achieve the basic functions of personal computer. In this paper, we use the raspberry pi 3b as the host computer. Its central processing unit is ARM Cortex-A53 and its graphics processing unit is Broadcom VideoCrore IV, which provides a guarantee for the operation of the algorithm. What is more, a fish eye camera is connected with the raspberry pi's 15-pin camera input for visual information. We select the stm32 as the slave computer. GY-85, which is a nine-axis inertial measurement unit, is connected with the stm32 board's GPIO interface. This enables the integration of INS information and odometer information. In this paper, we transmit the slave computer information to the host computer with a USB.

2.1 Installation of Software

Ubuntu is a desktop-based Linux open source system that supports x86, 64-bit and ppc frameworks. After downloading the corresponding vision of Ubuntu image file, we can install Ubuntu desktop environment for raspberry pi with the way of the u disk boot.

Robot Operating System is an open source secondary operating system and it includes two layers. The bottom layer is operating system layer, while the top layer is filled with all kinds of software packages to achieve the needs of customers. ROS provides the hardware abstraction and the data transmission between nodes. It makes the fuse of visual information and inertial information possible. Meanwhile, this platform can walk under the control. After configuring the Ubuntu depot and mirror source, we chose to install the full desktop vision of ROS. After that, we initialize the rosdep and configure the environment variables. Then, the ROS installation is completed and we can do our works with it.

OpenCV is an open source vision library which can be used on several systems, including Ubuntu. It provides some functions of image processing and image display, while it can work with ROS very well. We download the OpenCV source code from the official website and unzip it. Then we enter the folder and compile it with the cmake tool. After that, we need to add the directory into raspberry pi's environment variables. Finally, the installation of OpenCV is completed.

2.2 Remote Control of Raspberry Pi

Since this paper aims to design a real-time indoor positioning solution, so we need to display the real-time calculated moving trajectory. In this paper, we use a personal computer as the display screen. We use TightVNC to achieve the communication between personal computer and raspberry pi. After installing the TightVNC server and the TightVNC s client on personal computer and raspberry pi respectively, we can connect them with wireless LAN. Then, we can control raspberry pi remotely by Secure Shell, so that we can see the real-time moving trajectory on the personal computer.

2.3 Build of Mobile Car

We fix raspberry pi and stm32 board on the mobile car and fix the CSI camera and nine-axis inertial GY-85 on it. The camera is fixed upright to photograph the ceiling. The mobile car has a total of three wheels, including a caster wheel and two directional wheels. Two corresponding motors control the movement of two directional wheels, so that the mobile car can move forward, move backward, turn left and turn right. While the caster wheel plays a role of balance.

In this way, the whole experiment embedded platform is completed as it is shown in Fig. 1.

Fig. 1 Embedded platform on mobile car

We transmit the inertial information and odometer information from stm32 board to raspberry pi, and raspberry pi can get environment vision information from CSI camera. In final, we can calculate the mobile car's real-time position and display it on the personal computer.

3. Design of Positioning Algorithm

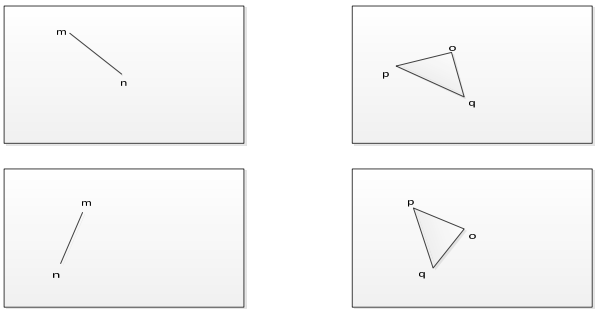
3.1 Inertial Positioning Methods

In this paper, we use wheel odometer and a nine-axis inertial measurement unit for positioning. The wheel odometer can calculate the real-time speed of mobile car and we mainly use z-axis gyroscope data because little car moves on horizontal ground. The z-axis gyroscope data can tells us the rotation angle when the mobile car turns. Therefore, we can judge the moving state and calculate the mobile car's moving trajectory with odometer and inertial measurement unit. However, there is a

cumulative error in this method, which is pretty serious in large experiment. It cannot meet our requirements for positioning accuracy.

3.2 Vision Positioning Methods

The objects on the ceiling are at the equal height, so its pixel coordinates have a great linear relationship with the vision coordinates of the mobile car. As the result, we select the objects on the ceiling as the source of our vision images. We use Oriented Fast and Rotated BRIEF algorithm to extract feature points of frames, so that we can calculate the camera movement between two frames. Due to the camera's large viewing angle, there are many points which are not on the ceiling in the frame. We need to exclude them. Besides, we want to exclude feature points which are mismatched. We use two rules to extract feature points we need from the whole points matched.



(a) (b)

Fig. 2 Two rules to extract the feature points

Firstly, as is shown in Fig. 2(a), we filter the matched points by comparing the lengths between any two pairs of matched feature points in two frames. If the difference of two lengths is bigger than a threshold, like 0.1 times of the length itself, then it means that the feature points are not on the ceiling or they are mismatched. In this way, a lot of points can be excluded. Secondly, as is shown in Fig. 2(b), we connect any three matching feature points in two frames individually to form two triangles. Then, we calculate the interior angles of these two triangles according to cosine theorem. If the difference of two corresponding angles is bigger than another certain threshold, like 0.03, it also means that the feature points are not on the ceiling or they are mismatched. In final, we can get server pairs of three matching points which are satisfied with above two conditions and we select any one of them. We can find the relationship between two matched feature points:

$$x_{2,n} = \cos H \cdot x_{1,n} + \sin H \cdot y_{1,n} + \frac{f}{d_x z_c} T_x \tag{1}$$

$$y_{2,n} = -\sin H \cdot x_{1,n} + \cos H \cdot y_{1,n} + \frac{f}{d_y z_c} T_y \tag{2}$$

The $x_{1,n}$ and $y_{1,n}$ stand the pixel coordinates in the first frame and the $x_{2,n}$ and $y_{2,n}$ stand the pixel coordinates in the second frame. The f is the focal length of camera and d_x, d_y is the size of frames. The H is the rotation angle while T_x, T_y is the displacement of mobile car. As is stated in paper [4], we can calculate the movement of the mobile car with this pair of points by least squares. However, this method is influenced easily by outside light. What's worse, sometimes we cannot find a pair of matched feature points. So, it is not satisfactory if we use the vision positioning alone.

3.3 Integration of Positioning Methods

As mentioned above, we are trying to find an algorithm that integrates the odometer, the inertial measurement unit and the vision camera to achieve a higher accuracy. We use a vector $[\theta, t_x, t_y]$ to express the movement of mobile car. Wherein, θ represents to the changed angle of the mobile car's forward direction. And t_x, t_y represent the displacement of forward direction and the displacement of perpendicular direction of forward. For example, the mobile car turn left by the angle of ninety, the

vector is equal to $[-\frac{\pi}{2}, 0, 0]$. In general, we can get two different sets of vectors through the inertial positioning methods and vision positioning methods. The mobile car has a total of three states. It includes stopping, going straight and rotation. We can judge the state by z-axis gyroscope. When the mobile car turn left or right, the z-axis gyroscope data is pretty large and when the car go straight, the z-axis gyroscope data is small. When the mobile car stop, the z-axis gyroscope data is approximate to zero. When the mobile car go straight, we use the inertial to calculate the moving trajectory. The calculation can be explained as the following equations:

$$ang\_v = gy\_z - noise \tag{3}$$

$$x\_pos_t = x\_pos_{t-1} + linear\_v \cdot \Delta t \cdot \cos(\theta_{t-1}) \tag{4}$$

$$y\_pos_t = y\_pos_{t-1} - linear\_v \cdot \Delta t \cdot \sin(\theta_{t-1}) \tag{5}$$

$$\theta_t = \theta_{t-1} + ang\_v \cdot \Delta t \tag{6}$$

The ang_v is the angular velocity of the mobile car, and the $linear\_v$ is the linear velocity of the mobile car. The gy_z stands for the z-axis gyroscope data and the noise is average of z-axis gyroscope data when the mobile car stops. The Δt means the inertial measurement unit's sampling time, which is 0.01 seconds in the experiment. The x_pos_t, y_pos_t and θ_t stand for the position and direction of mobile car at the time t. And when the mobile car turn left or right, we need to consider which method can be better. We photograph two frames before and after rotation according to the change of z-axis gyroscope data. Firstly, we calculate the moving trajectory with the equation (3) to equation (6) and we will decide whether to recalculate it later. For this rotation, we can get two sets of vectors, T_i and T_v. The T_i comes from inertial measurement unit and the T_v comes from matched feature points of two frames. This vector can be transformed into another vector which can describe the rotation and the movement of image. Therefore, we use these new vectors T_i' and T_v' to estimate the new frame with the frame before rotation. We compare the two new images with the frame after rotation respectively. We use $\frac{\sigma_{xy}}{\sqrt{\sigma_x^2 \sigma_y^2}}$ to define the similarity of two images. The σ_{xy} is the covariance of two images and σ_x^2, σ_y^2 are the variance of two images. If the image obtained by inertial measurement unit is more similar to the frame after rotation, we need not to do anything. The moving trajectory calculated with the equation (3) to equation (6) is pretty good. But while the image obtained by vision is more similar to the frame after rotation, we need to correct the θ and recalculate the moving trajectory from the time before rotation as the followings:

$$\theta_t' = \theta_t - slope \cdot (t - t_{before}) \tag{7}$$

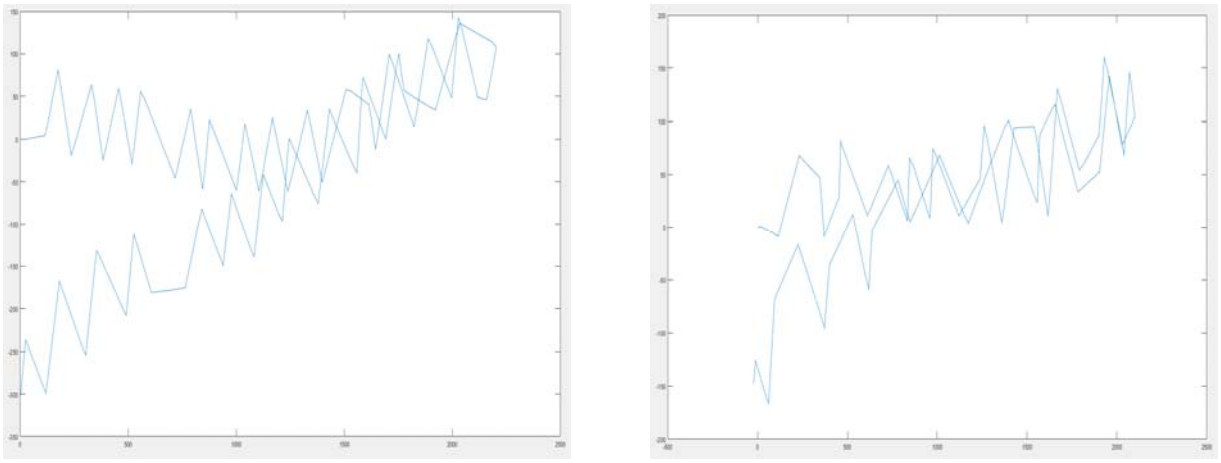
$$slope = \frac{\theta_i - \theta_v}{t_{after} - t_{before}} \tag{8}$$

In these equations, the θ_t is the former rotation angle and θ_t' is the angle corrected. The θ_i is the rotation angle calculated by inertial and θ_v is the rotation angle calculated by vision. The t_{before} and the t_{after} means the time before rotation and the time after rotation. In this way, we can recalculate the moving trajectory from the time before rotation to the time after rotation with the θ_t' and equation (3) to equation (6). And the new moving trajectory's accuracy is much higher.

4. Experiments

We have two experiments in different environments to verify the reliability of the algorithm. Firstly, we start the experiment in a corridor about 23 meters long. We move the mobile car from one side of corridor to the other side and return to the original place. The displacement between starting position and the ending position is considered to evaluate the goodness of algorithm. The result is

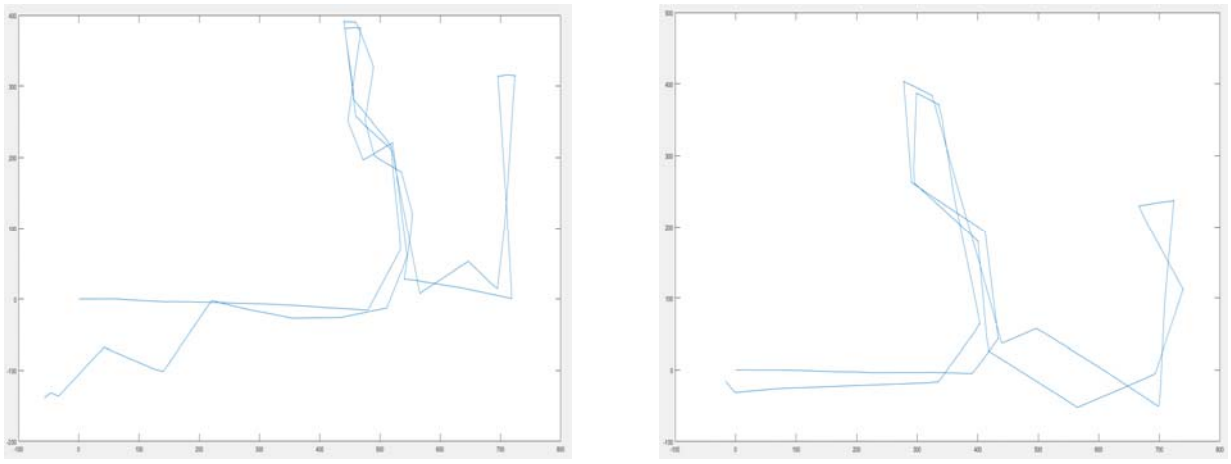
shown in Fig. 3. Fig.3 (a) shows the result of inertial and odometer and Fig.3 (b) is the result of the algorithm used inertial, odometer and camera. It can be calculated that the error of our algorithm is reduced to 48.2% of the original.



(a) (b)

Fig. 3 Results of a long corridor

We also have an experiment in two rooms and a corridor about 8 meters long. We move the mobile car from one side of corridor to the other side and return to the original place, entering two rooms through it. The result is shown in Fig. 4. Fig.4 (a) shows the result of inertial and odometer and Fig.4 (b) is the result of the algorithm used inertial, odometer and camera. And it can be calculated that the error of our algorithm is reduced to 14.8% of the original.



(a) (b)

Fig. 4 Results of a short corridor and two rooms

Besides, these experiments proves that our algorithm can be solved on embedded platform, which is shown in Fig. 5.

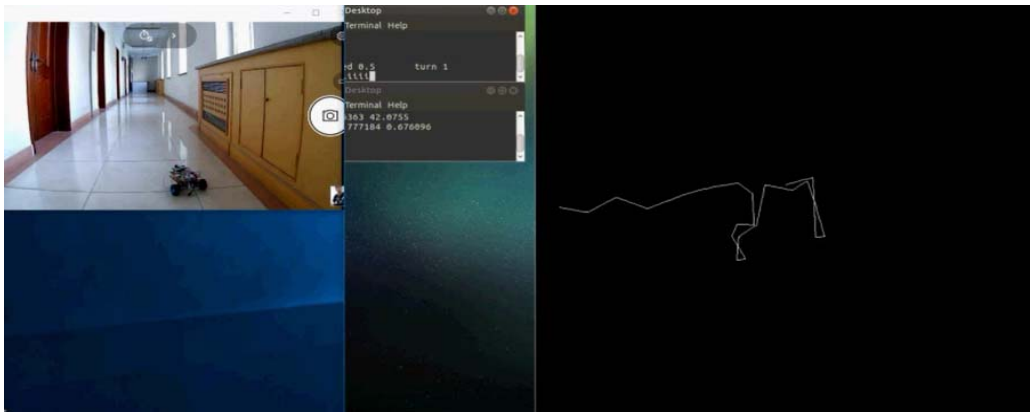


Fig. 5 Real-time trajectory

5. Conclusion

We get a higher accuracy with the integration of the inertial and vision. It is rarely influenced by outside light. Besides, we can calculate the real-time moving trajectory on embedded platform. Indoor positioning can be cheaper, more convenient and more accurate.

Acknowledgments

Grateful acknowledgment is made to my supervisor Liu Chang who gave me a lot of helps. And this work is supported by Key Research Program of Frontier Science, CAS, and Grant No. QYZDY-SSW-JSC037 and the National Natural Science Foundation of China (No. 61774157 and No. 81771388).

References

[1]. Lin Huican, Lv Qiang, Zhang Yang, et al. The sparse and dense VLAM: a survey [J]. Robot. Vol. 38 (2016) No. 5, p. 621-631.

[2]. Tateno K, Tombari F, Navab N. Real-time and scalable incremental segmentation on dense SLAM[C]// Proceedings of 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS), Hamburg, Germany, 2015: 4465-4472.

[3]. Schops T, Engel J, Cremers D. Semi-dense visual odometry for AR on a smartphone[C]// Proceedings of 2014 IEEE International Symposium on Mixed and Augmented Reality(ISMAR), Munich, Germany, 2014: 140-150.

[4]. Cao Tianyang, Cai Haoyuan, Fang Dongming, et al. Robot vision Localization system based on image content matching [J]. Opto-Electronic Engineering. Vol. 44 (2017) No. 5, p. 523-533.