GPU based real-time SLAM of six-legged robot

Zhang Xuehe, Li Ge, Liu Gangfeng, Zhao Jie, Hou ZhenXiu

Abstract

Vision and AHRS (attitude and heading reference system) sensors fusion strategy is prevalent in recent years for the legged robot's SLAM (Simultaneous Localization and Mapping), due to its low cost and effectiveness in the global positioning system. In this paper, a new adaptive estimation algorithm is proposed to achieve the robot SLAM by fusing binocular vision and AHRS sensors. A novel acceleration algorithm for SIFT implementation based on Compute Unified Device Architecture (CUDA) is presented to detect the matching feature points in 2D images. All the steps of SIFT were specifically distributed and implemented by CPU or GPU, according to the step's characteristics to make full use of computational resources. The registration of the 3D feature point cloud is performed by using the iterative closest point (ICP) algorithm. Our GPU-based SIFT implementation can run at the speed of 30 frames per second (fps) on most images with 900 × 750 resolution in the test. Compared to other methods, our algorithm is simple to implement and suitable for parallel processing. It can be easily integrated into mobile robot's tasks like navigation or object tracking, which need the real-time localization information. Experiments results showed that in the unknown indoor environments, the proposed algorithm's operation is stable and the positioning accuracy is high.

1. Introduction

The ability to work in an unexplored environment is crucial to the success of the autonomous robot's operation. The simultaneous localization and mapping (SLAM) is a technique used to build a map of the robot's surrounding while keeping track of its position. In recent years, significant attention has been paid to the development of the vision-based SLAM [1–3]. The reasons behind this popularity are the low price and low-power dissipation of cameras, large amounts of information contained in images as well as relatively simple mathematical models of cameras.

Most traditional mobile robot applied sonar [4] and laser ranging [5] to achieve SLAM, due to lower resolution of these distance sensors and the high uncertainty of observed data in complex environments makes them difficult to obtain the desired results. Monocular vision sensor usually combines sonar [6] or laser radar sensors [7] to achieve SLAM, since a single camera cannot get depth information by single image directly, it also need to calibrate the camera parameters and then obtain the three-dimensional coordinates of the feature points by additional method, the positioning accuracy is difficult to ensure.

Feature based simultaneous localization and mapping (SLAM) [8] has a long history. Modern approaches aim at matching all pixels, which require graphic card implementations due to the high computational load and partly rely on active sensors [9,10]. Scale-invariant feature transform (SIFT) [11] was an algorithm in computer vision to detect and describe local features in images. But the implementation of SIFT is complicated and time-consuming. Recently, accelerating SIFT algorithm on GPU and multicore CPU has achieved a remarkable progress [12]. Clipp B et al. [13] present a novel system for real-time, six degree of freedom visual simultaneous localization and mapping using a stereo camera as the only sensor. The system makes extensive use of parallelism both on the graphics processor and through multiple CPU threads. Seth Warn [14] has implemented parallel SIFT algorithm with OpenMP, and he also implemented Gaussian convolution using CUDA on GPU. Sequential implementations of SIFT are known to have high execution times. The open source sequential implementation SIFT++ [15] takes around 3.3 s on a 2.4 GHz processor for a speed of 0.31 fps for a 640 × 480 image. Sinha et al. [16] have used OpenGL/CG for their implementation on NVIDIA GeForce 7900 GTX card and reported a speed of 10 fps for a 640 × 480 video. Heymann [17] also implemented SIFT using OpenGL on GPU, and he organized the adjacent gray level pixels as RGBA pixels of a vector, which improved the execution speed of 17.24 fps for a 640 × 480 video. Aniruddha et al. [18] presents a parallel implementation of SIFT on a GPU, where obtain a speed of around 55 fps for a 640 × 480 image.
The binocular vision based real-time SLAM [19] only use a pair of cameras which placed in parallel to sense the surroundings. The internal and external parameters of the camera are obtained after calibrated. Three-dimensional coordinates of the feature points can be obtained directly from the left and right images by stereo matching algorithm. However, this method requires using the image information to restore robot motion information, the robust motion estimation is difficult to achieve [20]. Visual pose estimation can fail suddenly, due to bad lighting and texture conditions of the scene or fast movements that cause image blur. Supporting the camera measurements by a proprioceptive sensor like an AHRS should close the gaps where the image-based motion estimation fails and allow also for an increased update rate of the motion estimation. So this paper proposes a method combine the AHRS with binocular vision to achieve the SLAM of the robot. It operates in real-time at more than 30 frames per second by leveraging a combination of data parallel algorithms on the GPU, parallel execution of compute intensive operations and producer/consumer thread relationships that effectively use modern multi-core CPU architectures. Integrates the information [21] extract and analyze them in particular as follows.

2. GPU-based SIFT implementation

An overview over the proposed method is shown in Fig. 1. The binocular cameras are calibrated offline by the method proposed in [22], and then get the internal and external parameter of cameras. The rectified left and right images are used to detect feature points with the GPU-based accelerated SIFT operator. This produces typically several hundred of feature points. The SIFT detector is known to increase the ratio of correct matches by taking into account the scale of the difference of Gaussian (DoG) Scale-space define as,

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * l(x, y)$$

where \(\sigma\) and \(k\sigma\) are adjacent scales, and \(k\) is a constant multiplicative factor. The image scale space is expressed as discrete image pyramid which consists of several octaves, and the octave is comprised of several levels. The DoG pyramid is created by subtraction between adjacent images in Gaussian image scale space.

(B) Keypoint detection and localization

Once DoG images have been constructed, keypoint is identified as local minima/maxima of the DoG images. Each pixel in the DoG scale-space function.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

where \((x, y)\) represents image coordinates, \(\sigma\) is the scale factor and its size determines the degree of smoothing of the Difference of Gaussian (DoG) Scale-space define as,

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * l(x, y)$$

2.1. Algorithm overview of SIFT

Traditional SIFT algorithm include four main stages to extract feature points. Scale space construction, keypoint detection and localization, keypoint orientation assignment and keypoint descriptor. We introduce and analyze them in particular as follows.

(A) Scale space construction

Scale space defined as a function \(l(x, y, \sigma)\), which is produced from the convolution of variable-scale Gaussian, \(G(x, y, \sigma)\), with an input image, \(l(x, y)\):

$$l(x, y, \sigma) = G(x, y, \sigma) * l(x, y)$$

where \(*\) represents convolution and \(G(x, y, \sigma)\) represents Gaussian filter function.

The interpolation of keypoint is done using the quadratic Taylor expansion of the DoG scale-space function.

$$D(X) = D + \frac{\partial D}{\partial X} X + \frac{1}{2} \frac{\partial^2 D}{\partial X^2} X^2$$

Then, the location of the extreme point \(\hat{X}\) is determined by taking the derivative of this function with respect to \(X\) and setting it to zero.

$$\hat{X} = -\frac{\partial D}{\partial X}$$

$$D(\hat{X}) = D + \frac{1}{2} \frac{\partial^2 D}{\partial X^2} \hat{X}$$

The DoG function has a strong edge response. Therefore, in order to increase stability, Hessian matrix is used to eliminate the candidate keypoints that have poorly determined locations but have high edge responses.

$$H = \begin{pmatrix} \frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial x \partial y} \\ \frac{\partial^2 D}{\partial y \partial x} & \frac{\partial^2 D}{\partial y^2} \end{pmatrix}$$

$$\text{Tr}(H) = D_{xx} + D_{yy} = \alpha + \beta$$

$$\text{Det}(H) = D_{xx} D_{yy} - (D_{xy})^2 = \alpha \beta$$

where \(\alpha, \beta\) represent bigger eigenvalue, \(\beta\) represents smaller eigenvalue. Supposed that \(\alpha = \gamma \beta\), the ratio \(R\) can be defined as,

$$R = \frac{\text{Tr}(H)^2}{\text{Det}(H)} = \frac{(\alpha + \beta)^2}{\alpha \beta} = \frac{(r + 1)^2}{r}$$

Please cite this article as: Z. Xuehe et al., GPU based real-time SLAM of six-legged robot, Microprocessors and Microsystems (2015), http://dx.doi.org/10.1016/j.micropro.2015.10.008
It follows that, for some threshold eigenvalue ratio $r_{th}$, if $R$ for a candidate keypoint is larger than $(r_{th} + 1)^2/r_{th}$, the keypoint is poorly localized and hence rejected.

(C) Keypoint orientation assignment

Each keypoint is assigned one or more orientations based on gradient directions of local image. The keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation. For each image, $L(x, y)$, at this scale, the gradient magnitude, $m(x, y)$, and orientation, $\theta(x, y)$, is precomputed using pixel differences:

$$m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2}$$  \hspace{2cm} (11)

$$\theta(x, y) = \tan^{-1}\left(\frac{(L(x, y + 1) - L(x, y - 1))}{L(x + 1, y) - L(x - 1, y)}\right)$$  \hspace{2cm} (12)

(D) Keypoint descriptor

A set of orientation histograms are created on $4 \times 4$ pixel neighborhoods with 8 bins each. These histograms are computed from magnitude and orientation values of samples in a $16 \times 16$ region around the keypoint such that each histogram contains samples from a $4 \times 4$ sub-region of the original neighborhood region. The magnitudes are further weighted by a Gaussian function with equal to one half the width of the descriptor window. Since there are $4 \times 4 = 16$ histograms each with 8 bins, the vector has 128 elements. This vector is then normalized to unit length in order to enhance invariance to affine changes in illumination.

2.2. GPU implementation of SIFT

In this section, the whole details about SIFT acceleration method based on GPU is presented. We make full advantage of GPU’s abilities of parallel computation, float point computation, memory management, and give reasonable resource allocation to host (CPU) and device (CUDA). An overview of the steps in the GPU-based SIFT method’s implementation is shown as Fig. 2. The scale space construction, keypoint detection and localization, keypoint orientation assignment and keypoint descriptor are implemented in the device (CUDA).

2.2.1. DoG scale-space construction

Firstly, the image data is loaded from the host memory to the device memory and bound to texture memory. The advantage of that we chose texture memory are, the texture memory cache on the chip, it can reduce the memory request and provide more efficient memory bandwidth, to maintain a high performance of random read. On the other hand, when Gaussian filter is applied to 2D image array data, it is necessary to make judgment on image bounds. While the GPU is not good at executing conditional statements, using texture memory through property settings, we can deal with such problems automatically and efficiently.

The different scales of Gaussian kernel were uploaded to the constant memory of the device. According to the separability of Gaussian kernel function, the two-dimensional Gaussian convolution was decomposed into two one-dimensional Gaussian convolutions to achieve image filter [23]. Each image was divided into a series of image block with width $W$, height 1 and height $H$, width 1 as shown in Fig. 3. Each image block was processed by one thread block, and each thread in the thread blocks processed a row or a column of the image block. The sum of the horizontal and vertical Gaussian filter result is equal to the Gaussian filter result of the entire image. According to the layers, groups, Gaussian pyramid were assigned to a different block. Difference values of each pixel were processing in parallel and then sum the results of each block to accelerate the building of the Gaussian scale space pyramids.

2.2.2. Local keypoint detection and localization

Assume that Gaussian pyramid had 0 octaves and each octave had 5 layers. After the subtracted of the adjacent Gaussian image, we can get $S - 1$ difference-of-Gaussian (DoG) image. To detect the local maxima and minima of DoG each point is compared with the pixels of all its 26 neighbors. We firstly detect the extreme points in its 8 neighborhood, experiments showed that 95% candidate pixels can be eliminated. And then, we do extreme points detection in its 26 neighborhood for the kept pixels. The kernel functions can be circularly called $S - 3$ times to detect local keypoints. In order to reduce the judgment of wrap and improve the efficiency of the detection algorithm, each block processing $16 \times 16$ sized image block. If the width or the height of the image cannot exact divide by 16, filling the image edge with pixels which grey value are 0 as shown in Fig. 4. To a large extent, this method can improve calculation efficiency and save time. Because the low processing efficiency of device in branch and logical judgment, the extreme points which calculated preliminary were read back to the host to achieve precise positioning. Then the result of further selection was transferred back to device and saved in global memory.

2.2.3. Implementation of orientation assignment

Allocate the precise extreme point which has calculated in host to the device’s block. Each keypoint is processed in one block to calculate gradient orientation and magnitude of the pixel. Each block is divided into $16 \times 16$ threads and each thread processed one pixel.

2.2.4. Implementation of keypoint descriptor

We divide the square region around feature point whose edge is $12\sigma$ into $4 \times 4$ square sub-regions whose edge are $3\sigma$. An 8-orientation histogram is generated by calculating the contribution of gradient orientation of each pixel to the orientation histogram in a sub-region. So we can obtain $4 \times 4 \times 8$ values which compose the 128-dimensional vector. Considering that the histograms are data-independent from each other, we use one thread to process a sub-region. In the allocation strategy, there are 64 threads in a block to...
Block Thread
H=1
W
W=1
Thread
Block
(a) Horizontal Gaussian filter
(b) Vertical Gaussian filter
Fig. 3. The mapping diagram of Gaussian filter on GPU.

Fig. 4. Fill the image with 0 Gy value.

Fig. 5. Thread assignment of feature matching.

process four feature points. A thread is used to compute the weight of all the pixels in the sub-region and transfer eight pillars of histogram to global memory. Then the processing results of the 16 threads could generate a 128-dimensional feature vector.

2.2.5. Implementation of keypoint matching

The KD-tree search is more time-consuming in SIFT feature matching. The KD-tree is built in CPU and bound to the texture memory. Using Euclidean distance as a measure to find two nearest feature points in GPU, and then the data is stored in shared memory of each block. This can effectively improve the speed of data access. Each block contains 128 threads and each thread responsible for a feature point search. Thread assignment of feature matching is shown in Fig. 5. When the number of feature points less than 128, zero-paddling operation is adopted to meet the minimum computing needs of a block, reducing the overhead of judgment. When the number of feature points is much greater than 128, using multiple block to parallel computing.

2.3. Mismatching elimination

In traditional SIFT implementation, for a certain feature points in one image it must compare with all feature points in another image to find its corresponding. Each feature point has 128 dimensional data and the calculation amount is very huge. The runtime of the matching method is not suitable for real-time VSLAM.

Although there is aberration in the captured images, when we add the epipolar line constraint [24] the matching feature points are usually in the same epipolar line or nearby deviation of corresponding feature point P does not exceed ± δ pixels, δ = 3 based on experiment. The search scope of feature point matching only along epipolar line and it can reduce the amount of calculation effectively. In order to remove redundant match point, sequential consistency constraints and unique constraints were added. The sequential consistency constraint is the corresponding feature points in the same epipolar line of two images have the same order. Unique constraint requires that a feature in one stereo half-image be matched to, at most, one similar feature in the other half-image.

2.4. Experiment result of GPU based SIFT

To evaluate the performance of GPU-based SIFT method, tests are performed on ATI (Radeon HD 5450) and NVIDIA (GeForce GTX 430) graphics card. These tests showed an improvement of one order of magnitude in speed over a standard SIFT implementation. A 10X speedup over the CPU version is observed on the GPU-based graphics card. The feature points matching result of the real scene image is shown in Fig. 6. Fig. 7 compares the runtime between the CPU and GPU implementations for a range of image resolution. GPU-SIFT running on the NVIDIA GeForce GTX 430 could extract about 1000 SIFT features from streaming 900 x 750 resolution video at an average frame-rate of 30 Hz. Fig. 8 shows the running time of different steps in the GPU-based SIFT algorithm with different image resolutions. As shown in Fig. 9, the evaluation results shows that currently the NVIDIA graphic cards outperform the tested ATI graphics cards. This is due to the number of the stream processor and texture unit of NVIDIA graphic cards are superior to ATI. Our proposed algorithm for different graphics cards still have good acceleration effect, with the enhancement of graphics performance, acceleration performance of the proposed algorithm has been further reflected. The runtime of GPU-based SIFT method can meet the requirements of the six-legged robot’s real-time VSLAM.

We also compared the performance of the proposed GPU based SIFT method against SIFT [11], PCA-SIFT [25], ORB [26], Wu [27] and Acharya et al. [28] with the same conditions. As shown in Fig. 10, we found that the proposed GPU-based method outperformed the other...
3. Pose estimation implementation

We trigger the camera to capture images ahead of the robot in a certain time interval and the time interval determined by the speed of the robot. The initial correspondence step provides a set of corresponding points, whose 3-D position is known in respect to their camera coordinate system. The motion between two camera coordinate systems is described by a rotation $R$ and a translation $t$. The relation between the points of the previous and current frame ($C_i$ and $P_i$, with corresponding points having the same index $i$) can be written as,

$$C_i = RP_i + t, \quad \text{with} \quad i = 1, 2 \ldots n$$ (13)

Assume that feature points set which detected by the SIFT method are $S = \{s_i, i = 1, 2 \ldots k\}$ and $T = \{t_i, i = 1, 2 \ldots k\}$ respectively. For relative positions between the feature points does not change in the different perspective, so that they have the characteristics of spatial invariance. For example, the distance between feature points are not change in different view. According to the characteristic of feature space invariance, we can get a subset of the feature points obtained in the consecutive frame, which are named $P = \{p_i \in S, i = 1, 2 \ldots k\}$ and $Q = \{q_i \in T, i = 1, 2 \ldots k\}$ respectively. $p_i$ and $q_i$ are correspondence with each other. The essence of 3D point cloud data registration is coordinate transformation of the point cloud data which measured in different coordinate system. The key issue is to calculate the coordinate transformation parameters $R$ (rotation matrix) and $t$ (translation vector). The rotation matrix $R$ and translation vector $T$ should make the following objective function is
minimized:
\[
E = \sum_{i=1}^{n} \| q_i - (R_p + T) \|^2 \tag{14}
\]

For Eq. (14), we used the SVD matrix decomposition proposed by the literature [29], this method is easy to implement and the calculation results are more accurate.

For controlling, highly dynamic robotic systems require a delay free, robust and accurate system state estimate. The states of interest include position, attitude and velocity. This is especially essential for inherently unstable systems like six-legged robots. Visual odometry is generally very precise in measuring position and attitude due to the very high angular resolution of cameras. However, the quality or accuracy heavily depends on the texture as well as distance of features in the scene. In the worst case, binocular odometry may suddenly fail due to blurred images or textureless scene parts.

Another issue of visual odometry measurements is the time delay resulting from data processing delays. For control highly dynamic systems about six-legged robot, the time delay should not exceed 250 ms.

In order to increase the robustness and compensating for measurement time delays, it is necessary to combine visual odometry with other sensors. An attitude and heading reference system (AHRS), measuring angular rates and accelerations on three axes, is independent of the environment it is used in. AHRS is very suitable for legged robots as they are small and light weight. The sensor measurements are available at a high sampling rate and have a minimal time delay compared to the system dynamic of legged robots. Theoretically, the system attitude, position and velocity can be calculated by AHRS measurements only. Nevertheless, due to sensor noise and biases of AHRS the system states will drift after a short time and have, therefore, should be corrected using other sensor information. The binocular vision and AHRS complement each other to achieve real-time SLAM of the six-legged robot.

4. Experimental results and analysis

4.1. Testing platform

The VSLAM methods described in this paper are implemented on the experimental, insect-like hexapod robot, as shown in Fig. 11. Each of the six legs has three joints with three active degrees of freedom and a three-axis force sensor on each foot to sense the interaction force with terrain. All joints are driven by permanent magnet synchronous motors in combination with harmonic drive gears. Within each joint these are a motor angle sensor, a link side joint angle sensor as well as a joint torque sensor. In addition, it including an attitude and heading reference system (AHRS, three-axial gyroscopes, accelerometers and magnetometers, as shown in Fig. 12) to estimate the pose-in particular the instantaneous roll and pitch angles of the trunk. The size of the robot is 38.0 × 30.0 cm, and the mass is about 4.2 kg. The stereo vision camera is mounted in the front of the robot, such that it aims forward and downward.

Smart Eye-II stereo vision system is designed to obtain the three-dimensional information of scene, as show in Fig. 13. The transmission system of Smart Eye-II is constituted by the two joints, which are roll and pitch joint. The range of the scene image is determined by the turning range of the joint. The roll and acceleration speed of the joint determines the responsiveness of the stereo vision system, the rotary accuracy of the joints reflect the positioning accuracy of the stereo vision system. The range of roll and pitch joint is −60° to 60°, the angular velocity is 90°/s, positioning accuracy is 0.8°, and the optical center distance is 50–90 mm. The image resolution is 640 × 480.

4.2. Experiment result and analysis

Next, the VSLAM method is verified on a real robot. We do not have the motion capture system to measure the position of the robot, so the ground truth for the robot’s path is unknown. We build a man-made environment map in our laboratory and use the pre-set travel trajectory of the robot as ground truth of the robot’s path. The screenshot in the process of robot motion are shown in Fig. 14.

We evaluated the accuracy of our ego-motion estimation in two different experimental setups. In the first setup we only choose the stereo camera as sensor and measured the loop closure error. In the second setup we combine the stereo camera with AHRS to achieve pose estimation and trajectory generation of the robot. Considering the results of our performance tests, we chose for the tow experimental setups seven key frames and 300 feature points for real-time vision processing. We selected a test run with medium quality results from visual odometry with a final loop closure error of about 7 cm. Fig. 15 shows a 3D plot of the round trip trajectory with pure visual odometry in blue and reference in red. The reference path, visual odometry and AHRS fusion estimate of the 3D trajectory of second setup is shown in Fig. 16.

Fig. 17 depicts the absolute position errors of visual odometry and fusion estimate of visual odometry and AHRS compared to the reference path. Using seven key frames the data fusion algorithm can slightly improve the position error. Deactivating key frames, the

Please cite this article as: Z. Xuehe et al., GPU based real-time SLAM of six-legged robot, Microprocessors and Microsystems (2015), http://dx.doi.org/10.1016/j.micro.2015.10.008
visual odometry becomes more inaccurate which can easily be seen in Fig. 14. The calculated position from visual odometry drifts even though the device is standing still. Forcing interruptions of the visual odometry the fusion algorithm increases the quality of the ego-motion estimate significantly compared to visual odometry only.

5. Conclusions

The pose estimation of the robot plays an important role in the deliberative systems where the robot measures the state of the environment and plans its motion. The GPU-based real-time SLAM using stereo vision and AHRS as input sensors enable the HIT-II six-legged robot robust walking on rough terrain and mapping. The GPU-parallel method using CUDA achieves acceleration up to 10X contrast to traditional SIFT method and it can extract about 1000 SIFT features from images with 900 × 750 resolution at an average frame-rate of 30 Hz. The 6-DOF pose estimation and path generation of the robot is obtained from the 2D image features and its depth data by using ICP algorithm. Experiment results show that the path error is small enough to allow robust SLAM by using the stereo vision and AHRS. In spite of the feet slippages the robot can precisely estimate its pose and successfully recover from motion execution errors. Our method can either solve the monocular vision inaccuracy problem of feature point information obtained by special initialization method or avoid the enormous computation brought by binocular vision odometry using image information to restore movement. So far, the presented self-localization system is limited to small workspaces. This should be a topic of our further research with a possible experimental validation in outdoor scenarios and how to achieve SLAM in dynamic environment.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (NSFC) “Environment modeling and autonomous motion planning of six-legged robot” (No. 61473104). Support by Self-Planned Task (NO.SKLRS201410B) and (NO.SKLRS201501A02) of State Key Laboratory of Robotics and System (HIT). Support by National Magnetic Confinement Fusion Science Program “Multi-Purpose Remote Handling System with Large-Scale Heavy Load Arm” (2012GB102004). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References


Please cite this article as: Z. Xuehe et al., GPU based real-time SLAM of six-legged robot, Microprocessors and Microsystems (2015), http://dx.doi.org/10.1016/j.micpro.2015.10.008
Zhang Xuehe received B.S., M.S. degrees from Liaoning University of science and technology in 2008 and 2011 respectively, all in computer science and technology. From September 2011 to now, he is a PhD student of Harbin Institute of Technology. His research interests include the joint area of computer graphics and vision, including 3D shape reconstruction from multiple views, computational photography, and Stereo vision navigation and path planning of the six-legged robot.

Liu Gangfeng received Ph.D. degrees from Harbin Institute of Technology in 2010, in Mechanical and Electronic Engineering. Since September 2010, he has been with the School of Mechanical and Electronic Engineering, Harbin Institute of Technology, where he is a lecturer. His research interests include the robotic tele-operation technology and space manipulator technology.

Zhao Jie received B.S.,M.S., and Ph.D. degrees from Harbin Institute of Technology in 1990, 1993, and 1996, respectively, all in Mechanical and Electronic Engineering. Distinguished professor of the cheung kong scholars programme of ministry education. Leader of advanced manufacturing intelligent robot of the national ‘twelfth five-year’ and ‘863 plan’. His research interests include the multi-sensor system integration and control technology, robotic tele-operation technology based on Internet network.

Hou Zhenxiu received B.S.,M.S., and Ph.D. degrees from Harbin Institute of Technology in 1982, 1991, and 2006, respectively, all in Mechanical and Electronic Engineering. Since September 1982, she has been with the School of Mechanical and Electronic Engineering, Harbin Institute of Technology, where he is an assistant professor. His research interests include the multi-sensor system integration and control technology, robotic tele-operation technology based on Internet network.

Please cite this article as: Z. Xuehe et al., GPU based real-time SLAM of six-legged robot, Microprocessors and Microsystems (2015), http://dx.doi.org/10.1016/j.micpro.2015.10.008