SIFT Based Monocular SLAM with GPU Accelerated

Tonghui Wang, Guoyun Lv(✉), Shikai Wang, Haili Li, and Baicen Lu

School of Electronics and Information, Northwestern Polytechnical University, Xi'an, China
nwpuwth@qq.com, l78510887@qq.com, l748710855@qq.com, 735110965@qq.com, 992159044@qq.com

Abstract. With the rapid development of computer vision technology, 3D Reconstruction based on monocular SLAM (Simultaneous Localization and Mapping) has got more and more attention for its simple requirements, low cost, easy to implement, convenient to carry. ORB-SLAM is a kind of monocular SLAM method based on feature point. ORB feature can meet the real-time requirements for SLAM, but it does not have scale invariance. In this paper, we proposed a monocular SIFT-SLAM, in which a SIFT (Scale Invariant Feature Transform) algorithm based on GPU is used to replace the ORB algorithm, to implement 3D Reconstruction. We show the experiment result of SIFT-SLAM in this paper, which gets some improvement.

Keywords: SLAM · Monocular · SIFT · GPU · 3D Reconstruction

1 Introduction

1.1 Monocular Visual SLAM

In recent years, SLAM based 3D Reconstruction method has become a hot issue in the field of computer vision [1, 2]. SLAM [3] is a process in which the mobile robot senses the surrounding environment through the sensor carried by itself, and uses the information obtained by the perception to carry on the self-localization process. Compared with the expensive and complex shortcomings such as laser sensors, ordinary visual sensors have advantages such as a wide range of measurement, rich information collection, cost-effective, versatile, convenient to carry and so on. In addition, the visual SLAM is real-time system. Based on these advantages monocular vision SLAM technology becomes the direction of SLAM development [3, 10–12].

At present, the vision-based SLAM technique is divided into two types: feature-based method and direct method. The direct method uses all the pixels in the image for map reconstruction based on the gray scale invariant assumption of the object feature point, so the extraction of the feature points is avoided and reduce the computational complexity, and ensure the real-time performance of the algorithm. The main representative is LSD-SLAM (Large-Scale Direct Monocular SLAM) [4] proposed by Jakob Engle and others of the University of Munich, Germany. The feature point method considers the feature point as the fixed point of the fixed three-dimensional space,

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2018
B. Li et al. (Eds.): ChinaCom 2017, LNICST 237, pp. 13–22, 2018.
https://doi.org/10.1007/978-3-319-78139-6_2

calculates the three-dimensional coordinate transformation matrix of the feature point according to the matching of different image feature points, and implement the object reconstruction, the representative method is the ORB-SLAM [5] algorithm proposed by Mur-Artal et al. At present, the visual SLAM algorithm based on feature point matching is still the mainstream research direction of SLAM field. But the traditional feature point method still has obvious shortcomings. ORB (Oriented BRIEF) [6] feature point, which uses FAST as the feature point and detection operator BRIEF as a feature descriptor, has the advantages of rotation invariance, affine invariance, translation invariance. But ORB does not have good robustness for noise and scale invariance. In 2004, David proposed the SIFT (Scale Invariant Feature Transform) [7] feature point, which has translation invariance, rotation invariance, scale invariance, affine invariance, and the robustness for noise and light are better. However, the SIFT algorithm has high complexity and large computational complexity, which leads to poor algorithm performance and cannot be directly used in real-time visual SLAM system.

This paper presents the calculation of SIFT algorithm on the GPU platform [11] to solve the real-time problem of SIFT algorithm. The accelerated SIFT algorithm is used to replace the ORB algorithm in ORB-SLAM system, to realize the SIFT-SLAM system based on monocular vision. This paper gives the results of the accelerated SIFT algorithm, and evaluates the feasibility and effectiveness of the SIFT-SLAM system.

1.2 GPU and CUDA

The rapid development of semiconductor technology, make CPU and GPU with faster and faster computing speed. But the CPU and GPU have their own unique advantages. The difference between CPU and GPU are as Fig. 1.

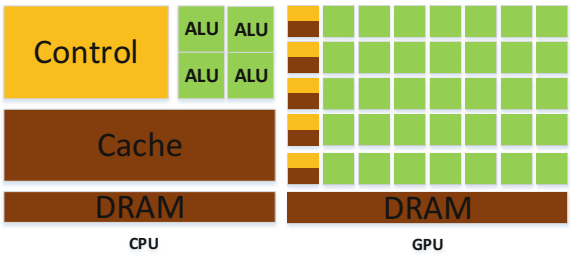

Fig. 1. CPU and GPU structure diagram. CPU has bigger Control area and smaller ALU area, which makes CPU be more capable in logic control. GPU has more ALU which means GPU has faster computing speed.

The CPU facilitates a large number of programs including loops, branches, logical decisions, and complex instructions; the GPU is more advantageous for processing with programs that have high parallelism, large data volumes, low data coupling, high computational density, and less interaction with the CPU.

NVIDIA has released a CUDA parallel computing platform that makes heterogeneous programming based on CPU and GPU easy to implement. The CPU is called Host, and GPU is called Device. In this heterogeneous model, the CPU is responsible for the logic control and serial computing. The GPU is responsible for the highly parallel computing of large-scale data in the system A part of the parallel processing of a CUDA program is done by the kernel function. CUDA threads are divided into three levels: grid, block, thread. Every thread of every block in every grid has a unique thread index, which we named it as threadIdx. The threadIdx ensures that each thread can read the corresponding data in the memory space to ensure that the parallelization of the normal calculation. CUDA reduces the use of loop structures by parallel computation, as a result this saves a lot of time.

2 ORB-SLAM Framework

ORB-SLAM can run on computer in real time with good system robustness. The overall framework of the ORB-SLAM system is shown in Fig. 2. The whole system is divided into three parts:

Tracking. The tracking part mainly extracts the ORB feature from the image, performs the pose estimation according to the previous frame, or initializes the pose through the global positioning, optimizes the position, and then confirm the new key frame according to some rules.

Local Mapping. This part mainly completes the local map construction, including the insertion of the new key frame, verifies and filters the newly generated map points, and generates the correct map points. After that, the local Bundle Adjustment (BA) is used to filter the inserted key frame and remove redundant key frames.

Loop Closing. This part is divided into two parts, loop closing detection and loop closing correction. Loop closing detection is first performed using a Bag of Words [8] and then simulated by Sim3. Loop closing correction, mainly closed-loop fusion and g2o (general graph optimization) [9] optimization.

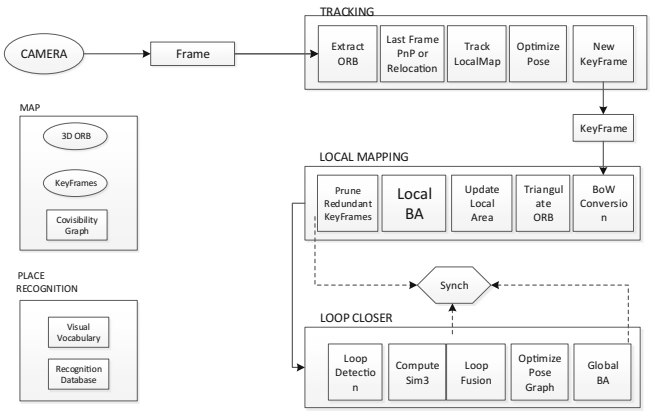

Fig. 2. The framework of ORB-SLAM.

From the Fig. 2, it can be seen that ORB-SLAM extracted the ORB feature points from the picture taken by the camera. In this paper, we use the SFIT algorithm after GPU acceleration to extract the feature points, so we can get a real-time SLAM system under the premise of a good 3D Reconstruction result.

3 Accelerated SIFT Algorithm

3.1 SIFT Algorithm

David G. Lowe proposed the Scale-Invariant Feature in 1999 to carry out object recognition and image matching. In 2004, he proposed the Scale Invariant Feature Transform (SIFT) algorithm for development and refinement.

The SIFT algorithm contains four parts: the scale space is built and the extreme points are detected, the feature points are selected and positioned, and the direction values are determined for the feature points (Fig. 3).

First, the original image is layered by using Gaussian filters of different scales, and then the adjacent Gaussian filtered image is subtracted to obtain the difference of Gaussian (DoG) pyramid. Then we select the extreme point from the scale space. The feature points with lower contrast and the feature points located at the edge position are removed, leaving the remaining feature points which meet the threshold requirements. In the discrete function extremum points can represent the mathematical characteristics of the discrete function. The maximum gradient modulus value of each pixel at the feature point and its neighborhood will be the main direction of the feature point.

Select a rectangular block pixels around a key-point, and divided it into $16 = 4 \times 4$ sub regions as suggested in Lowe's paper [7]. We draw the cumulative value for each gradient direction, forming a key point, which include 8-way vector information. As a result, one feature point can generate a $4 \times 4 \times 8$ dimensional data, which means, a total of 128-dimensional data to form a 128-dimensional SIFT feature vector.

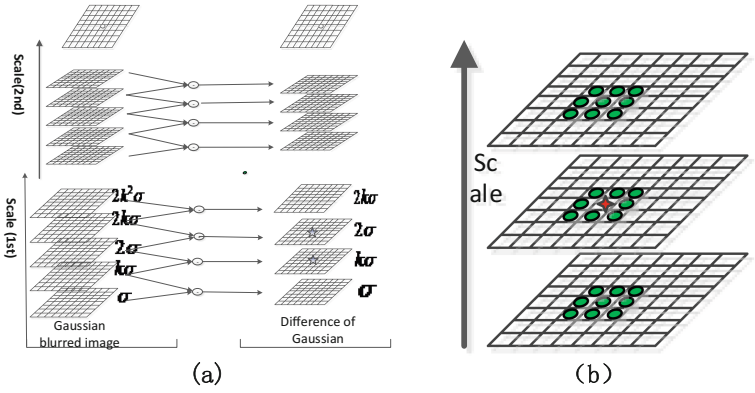

Fig. 3. (a) is the figure of Gaussian and Difference of Gaussian. (b) is extreme point value detection in Gaussian scale-space, the red point means the feature point, the green point represents the pixels wait to be compared. (Color figure online)

3.2 Accelerate SIFT with GPU

Create Difference of Gaussian Pyramid. The SIFT algorithm creates the scale space by Gaussian blur, uses the Gaussian function to compute the fuzzy template. Then the template is used to do convolution with the original image.
The equation of the two - dimensional Gaussian function $G(x, y)$ is as follows:

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

m represents the width of image and n represents the height of image, σ represents deviation of normal distribution. x, y represent the pixel coordinates.

When we calculate the DoG pyramid, each blurred image obtained is based on the original image to take different σ values to achieve. The coordinates of each layer in the octave are calculated as follows:

$$\sigma(s) = \sqrt{(k^m\sigma_0)^2 - (k^{m-1}\sigma_0)^2} \tag{2}$$

m represents the index of an interval in an octave, σ_0 represents the initial value of σ.

So we can put the process of creating a scale space into the GPU. Assuming that the Gaussian pyramid has N octaves, each octave contains M intervals, then in the GPU the kernel will create $N \times M$ threads, giving each thread a different σ value. Through a parallel calculation, the Gaussian blurred images can be calculated. Afterwards adjacent image is subtracted to obtain a difference of Gaussian pyramid.

Extreme Point Detection. In the Gaussian scale-space, each pixel is compared with pixels in its 3 * 3 neighborhood at its interval and the adjacent interval. When it is an extreme value, the feature point is stored.

For a difference of Gaussian pyramid with N octaves, each group of M internals, the number of pixels between the different octave are different and independent with each other. The kernel function is performed separately from the feature point detection between the octaves. The kernel function performs the feature point calculation of a set of Gaussian blurred images at a time, and loop N times to complete all the calculations.

Define Direction for Key-Point. In order to make the descriptor have rotational invariance, it is necessary to use the local feature of the image to assign a reference direction for each key-point. Using the method of calculating image gradient to find the stable direction of local feature. For the key-points detected in the DOG pyramid, the gradient and direction distribution of the pixels in the neighborhood window of the Gaussian pyramid image are calculated.

The formulas used to calculate the modulus value and direction of gradient as follows:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \tag{3}$$

$$\theta(x, y) = \tan^{-1}(L(x, y+1) - L(x, y-1))/(L(x+1, y) - L(x-1, y)) \tag{4}$$

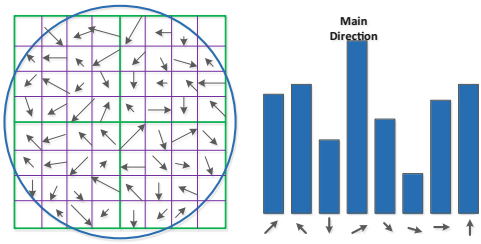

Fig. 4. Gradient direction of Key-point and histogram of gradient modulus value (For simplicity, only eight bins are show in the histogram).

$m(x, y)$ and $\theta(x, y)$ represent the modulus and direction of the gradient, respectively. L represents the scale space of the key points.

After completing the gradient calculation of the key-points, we use the histogram to measure the gradient and direction of the pixels in the neighborhood. The gradient histogram divides the range of 0 to 360° into 36 columns (bins), 10° per column. As shown in Fig. 4, the peak direction of the histogram represents the main direction of the key-point [7].

The peak of the direction histogram represents the direction of the neighborhood gradient at the key-point, and the maximum value in the histogram is taken as the main direction of the key-point. According to the calculation process, the gradient direction and the gradient modulus of all the points in the neighborhood of the key-point are put into GPU. Assuming there are n points in the neighborhood, the GPU kernel function create n CUDA threads, and executes the calculation process. And then the search of main direction and auxiliary direction is calculated in CPU.

Generates Feature Point Descriptors. Through the above steps, for each key, there are three information: location, scale and direction. The next step is to create a descriptor for each key, and describes the key point with a set of vectors so that it does not change with various changes, such as changes in light, changes in perspective, and so on.

We rotate the coordinate axis to the main direction of the key point to make the rotation invariance. Then we draw the cumulative value for each gradient direction, forming a key point, which include 8-way vector information. As a result, one feature point can generate a $4 \times 4 \times 8$ dimensional data, which means, a total of 128-dimensional data to form a 128-dimensional SIFT feature vector (Fig. 5).

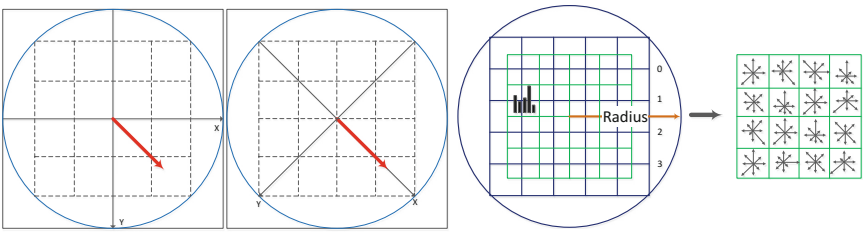

Fig. 5. Rotate the axis towards the main direction of key-point and the descriptor of SIFT feature

In the calculation, rotating the axis is carried out in the CPU, and when the eight degrees of freedom in the 16 sub-regions are calculated, the kernel function in the GPU is opened with 128 threads, and the feature points are generated together.

4 Experiment and Results

4.1 Experiment Platform

In this test the hardware platform we used is the Intel Core I7-6700, GPU for the NVIDIA GTX 1060. The specific test environment is shown in Table 1.

Table 1. Parameters of experiment platform

Class	Contents
System	Linux Ubuntu 14.04 and 64
CPU	Intel(R) Core(TM) i7-6700 CPU @ 2.40 GHz
RAM	16 GB
GPU	NVIDIA GeForce GTX 1060
CUDA version	8.0
OpenCV version	3.0

This section selects three sets of images with different resolutions and scales in Fig. 6(a), (b) and (c) for experiments.

We use three different images of different images to experiment, Fig. 6(a), (b) and (c) three different resolution images, the second image of each group has angle translation and scale transformation relative to the first image. Respectively, with the ORB algorithm in OpenCV, the SIFT algorithm in OpenCV, the accelerated GPU-SIFT algorithm to test 10 times, take the average. The final results are shown in Table 2 and Fig. 6(d).

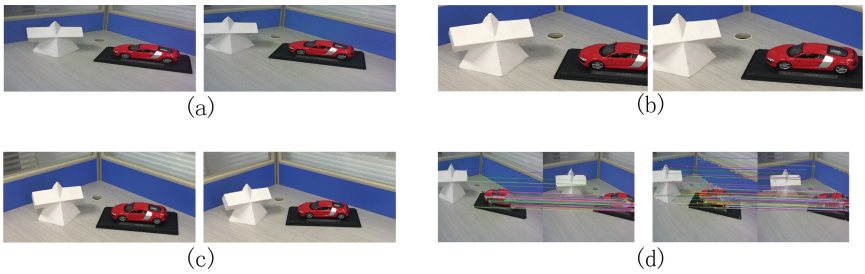

Fig. 6. (a) Image resolution 640 × 480; (b) Image resolution 1280 × 720; (c) Image resolution 1920 × 1080; (d) is the result of ORB algorithm and GPU-SIFT algorithm.

The Time Cost above is used to extract the feature points of two pictures and match them. From the experimental results, we get the follow conclusions:

Conclusion A. Compared with the original SIFT algorithm, the GPU-accelerated SIFT algorithm has obvious acceleration effect at the speed of the feature points extraction. For 640 × 480 resolution images, the time cost is reduced by nearly 20 times, while for 1920 × 1080 resolution images, the time cost is reduced by nearly 70 times. From this we can see that when the GPU is dealing with parallel computation of large data, the acceleration effect is very obvious.

Conclusion B. Compared with the traditional ORB algorithm, GPU-accelerated SIFT algorithm in the feature point extraction speed makes up for their own shortcomings, timeliness problems have been resolved. Besides SIFT algorithm gets more feature points than ORB algorithm, which means the match will be more accurate.

Conclusion C. The above experimental results are not considered in the CPU-GPU heterogeneous programming platform, the time spent with the data transfer in the host memory and graphics memory. So in the actual SIFT-SLAM system, the use of time-consuming will increase (Fig. 7).

Table 2. Experiments based on NVIDIA GeForce GTX 1060 and result

Image resolution	Algorithm	Number of feature		Match feature	Time cost
640 × 480	ORB	500	500	200	21 ms
	SIFT	359	409	132	270 ms
	GPU-SIFT	346	371	128	14 ms
1280 × 720	ORB	500	500	175	65 ms
	SIFT	1273	1403	609	878 ms
	GPU-SIFT	1187	1276	593	18.8 ms
1920 × 1080	ORB	500	500	51	114 ms
	SIFT	2206	2361	126	1870 ms
	GPU-SIFT	1962	2289	107	26.9 ms

Fig. 7. The left picture shows the source picture. And the right one is the result of SIFT-SLAM algorithm.

4.2 SIFT-SLAM Experiment

In this experiment, we collected the video sequence through the camera as the data source, which was tested in the monocular ORB-SLAM system and monocular SIFT-SLAM system respectively.

In the experiment, we use ORB-SLAM and SIFT-SLAM to rebuild the building. As the result, the SIFT feature gets more feature points than ORB, and we get a clearer silhouette with SIFT-SLAM.

So we get the conclusion that SIFT feature is more suitable for 3D Reconstruction in feature-based slam system than ORB feature. But in the experiment, SIFT-SLAM is a little slow than ORB-SLAM, which we still need to optimize.

5 Conclusions

In this paper, we proposed a method, using the GPU to accelerate SIFT algorithm for real-time calculations. Combined with mature ORB-SLAM, we use accelerated SIFT algorithm to replace ORB algorithm, and we propose a SIFT based monocular SLAM system. As experiment results show, the accelerated SIFT algorithm can make feature detection run on real-time. Besides SIFT feature has greater advantages in scale invariance and robustness compared with ORB algorithm. The SIFT-SLAM monocular system generate a spare point cloud map, which is enough for simple 3D Reconstruction.

Acknowledgments. This work is sponsored by the Seed Foundation of Innovation and Creation for Graduate Students in Northwestern Polytechnical University (No. Z2017139).

References

1. Smith, R., Self, M., Chessman, P.: Estimating uncertain spatial relationships in robotics. In: Proceedings of the Uncertainty in Artificial Intelligence (1988)
2. Davison, A.J., Nobuyuki, K.: 3D simultaneous localization and map building using active vision for a robot moving on undulating terrain. In: Proceedings of the IEEE International Conference on Computer Vision and Recognization, Huwaii, pp. 384–391 (2001)
3. Andersen, D.R., Cuykendall, R., Regan, J.J.: SLAM - vectorized calculation of refraction and reflection for a Gaussian beam at a nonlinear interface in the presence of a diffusive Kerr-like nonlinearity. Comput. Phys. Commun. **48**(2), 255–264 (1988)
4. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: large-scale direct monocular SLAM. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8690, pp. 834–849. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10605-2_54
5. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Trans. Robot. **31**(5), 1147–1163 (2015)
6. Rublee, E., Rabaud, V., Konolige, K.: ORB: an efficient alternative to SIFT or SURF. In: Proceedings, vol. 58, no. 11, pp. 2564–2571 (2011)
7. Lowe, D.G.: Distinctive image features from scale-invariant key-points. J. Comput. Vis. **2** (60), 91–110 (2004)

8. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, NY, USA, pp. 2161–2168 (2006)
9. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g2o: a general framework for graph optimization. In: International Conference on Robotics and Automation (ICRA) (2011)
10. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, pp. 225–234 (2007)
11. Klein, G., Murray, D.: Improving the agility of keyframe-based SLAM. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008. LNCS, vol. 5303, pp. 802–815. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88688-4_59
12. Mur Artal, R., Tardos, J.D.: Fast relocalisastion and loop closing in keyframe-based SLAM. In: Proceedings of the IEEE International Conference on Robotics and Automation, New Orleans, LA, pp. 846–853 (2014)