

Speed Optimization of UAV Vehicle Tracking Algorithm

Yanming Xu, Wei Pei, Yongying Zhu, Mingyu Lu, Lei Wu

▶ **To cite this version:**

Yanming Xu, Wei Pei, Yongying Zhu, Mingyu Lu, Lei Wu. Speed Optimization of UAV Vehicle Tracking Algorithm. 2nd International Conference on Intelligence Science (ICIS), Oct 2017, Shanghai, China. pp.390-399, 10.1007/978-3-319-68121-4_42 . hal-01820913

HAL Id: hal-01820913

https://inria.hal.science/hal-01820913

Submitted on 22 Jun 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Speed Optimization of UAV Vehicle Tracking Algorithm

Yanming Xu[1], Wei Pei[2], Yongying Zhu[3], Mingyu Lu[1], Lei Wu[1]

[1] Information Science and Technology College, Dalian Maritime University, Dalian, Liaoning Province, China
[2] Environmental Science and Engineering College, Dalian Maritime University, Dalian, Liaoning Province, China
[3] Ocean and Civil Engineering department, Dalian Ocean University, Dalian, Liaoning Province, China
peiweidl@gmail.com

Abstract. An improved method for vehicle tracking in UAV-captured video is introduced in this paper, this method based on GPU (Graphics Processing Units) acceleration which have efficient parallel computing powers. Today, vehicle tracking in UAV videos has many meaningful applications in computer vision, such as traffic management and illegal vehicle tracking. However, the increased computational complexity makes it difficult to be used in real-time tracking system. To overcome the limitation of tracking speed and meet the highly intensive calculation required, we perform image processing with GPUs and adopt an optimized structured output support vector machine (SVM) to online learning. The proposed approach is evaluated in UAV videos taken from DJI Matrice 100 drone and the corresponding conclusions are received.

Keywords: vehicle tracking; speed optimization; GPU; UAV

1 Introduction

With the rapid development of unmanned aerial vehicle technology (UAV), the detection and tracking of moving vehicles in UAV videos has been paid more attention. In military, UAV can be used in casualty rescue, battlefield investigation and surveillance, medicine material transport, damage assessment and so on [1], [2], etc. UAV object tracking has also been applied to many civilian areas. Traffic police department can conduct the identification of illegal vehicle behavior in a remote distance which greatly saves money and material resources [3]. Including Forest fire, marine pollution monitoring and other highly destructive events can also be prevented by UAV detection. The most widespread use of agricultural UAV is medicine spraying [4]. Drone object tracking is a significant research which involves many disciplines such as artificial intelligence, image processing, machine learning.

However, target tracking of UAV remains as a challenge because of the relatively lower video resolution, non-uniform illumination, object occlusion, shape deformation, irregular rotation, small-sized target, drone vibration and so on [5]. In order to track a moving vehicle with a UAV, the following points should be implemented in the

tracking algorithm. Firstly, the accuracy of the result should be guaranteed in object tracking. Algorithm should have the ability to identify the tracking target from a complex, noise-filled background. Secondly, the tracking algorithm is supposed to keep a certain degree of robustness when the moving target is partially or completely obscured. It means that an automatic re-detection process platform should be established to keep tracking the object. Lastly, the most important thing one has to take into consideration is the processing speed. Videos transferred from a real UAV camera ought to be processed at high speed. Many works have been done to reduce the processing time, but the processing consumption increases exponentially with a significant improvement in UAV video resolution. A single CPU processing has been unable to meet the requirements for speed. With the rapid development of GPU hardware and its programming model, using the GPU's massive high-speed parallel processing mechanism to accelerate the machine vision algorithm has become an inevitable trend. This paper strives to improve the processing speed under the premise of high precision on an improved tracking algorithm [6] based on Struck [7] by GPU acceleration.

2 Related Works

Large quantities of algorithms have been proposed in recent years, but they all have significant limitations. G. Mattyus pointed out that the rapid movement of the UAV camera which resulting in the distribution of samples can be used to express characteristics are too little, so by normalizing the European distance to separate the front and back points, and then get the moving target binary image [8]. But the calculations is too complex to lead to a slow tracking process. This problem can be solved by CUDA Parallel operation as described in this paper. Based on background difference, an improved method of moving object detection and tracking is proposed, and the target is identified with the background model to construct the deviation among different image frames [9]. Considering the relationship between the target height and the shadow scale, GV.Reilly performed shadow removal and target tracking by combining wavelet features with SVM features in 2013 [10]. The background subtraction was used to overcome the difficulties of scene constantly shifts [11], [12], etc. In order to achieve higher accuracy and get a relatively good tracking performance, many algorithms involve large feature vectors which bring a significant impact on the speed of the tracker.

The purpose of this paper is to accelerate the tracking process By CUDA computation. CUDA is a general purpose parallel computing platform and programming model that it leverages the parallel compute engine in NVIDIA GPUs to solve many complex computational problems in a more efficient way. Using a machine learning method to choose the best sparse matrix representation model on GPU [13], the target tracking algorithm was implemented on the CUDA platform with the GPU's ability of parallel computing [14], [15], etc. Pei [6] proposed an improved tracking method based on Struck [7], the extended affine transformation motion estimation method had been adopted to automatically adjust the tracking window's scale and rotation. Compared with the original Struck, the tracking accuracy has been enhanced,

however, the number of frames processed per second have been dropped. The meaningful contribution in this paper is that all those tracking models are ported to the GPU and keep detection and tracking via CUDA parallel calculations.

3 System Achievement

In this paper, GPU's powerful parallel computing capabilities are exploited to accelerate the speed of improved tracking algorithm. We use Gaussian/Haar-like combination to reach an acceptable compromise between processing speed and tracking accuracy. An online structured output SVM learning framework has been used to incorporate multiple types of image features and kernels. In this section, some key points will be introduced briefly.

3.1 CUDA Programming Model

According to CUDA's model architecture, the task of the CPU is to perform serial operation such as data transmission and logic operations. Originally graphics data processing consumed lots of resources now is handled by GPU parallel computing. CUDA not only completes the traditional image rendering work, but also achieves good performance of complex operation. A complete CUDA program includes the serial processing program in host (CPU) and the device's (GPU) parallel processing code. Code which runs on the GPU is called the CUDA kernel function. To complete respective subtasks of each model, the CUDA threads run on a physically separate device (GPU) which plays as a medium to the host running the C code. The host code will calls the parallel code running on the device side when needed [16].

3.2 Haar-like Feature Extraction

Papageorgiou first proposed Haar-like features and applied it to face description [19]. Pixel-based face detection algorithm has a high computational cost, but the Haar-Like feature based on block reduces processing costs. This paper uses six kinds of Haar-like features (feature template), as shown in Fig. 1. The reason why we choose this six features is it basically cover all the features. The feature value is that the sum of black pixels gray value subtracts the sum of white pixels gray value.

The main idea of integral images is to form a rectangular region from the starting point to each point and let the sum of pixels value in each rectangular region as an element of an array stored in memory. When you need to calculate the pixels value of a region, you can index the array elements, Instead of recalculating the pixels of this region, thus the computation will be speeded up. Integral image algorithm is a matrix representation method which can describe the global image information. The principle is that the element value $ii(x, y)$ at location (x, y) in integral image is the sum of all pixels above from the upper left corner to (x, y) direction in original image f. The recurrence formulae are as follows.

$$\mathrm{s}(x, y) = \mathrm{s}(x, y-1) + i(x, y)\ . \tag{1}$$

$$ii(x, y) = ii(x - 1, y) + s(x, y) . \quad (2)$$

Where $s(x, y)$ is the cumulative row sum with the initialization of $s(x, -1) = 0$. $ii(-1, y) = 0$ and $ii(x, y)$ is an integral image. Scanning the image and when the image reaches the lower right corner of the pixel, the integral image is over.

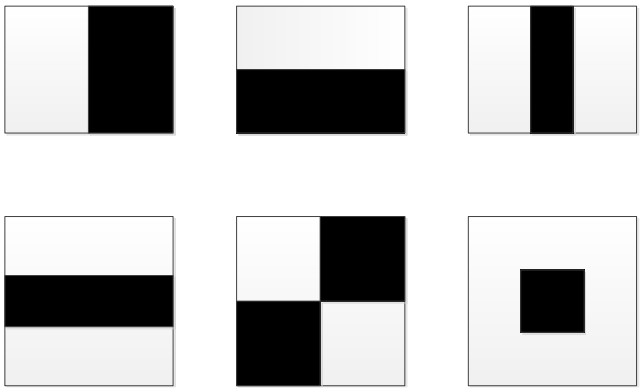


Fig. 1. Six Box Features Template for Extracting Haar-like Feature.

3.3 Structured Output SVM Optimization

Support vector machine (SVM) is a statistical learning theory based on Structural Risk Minimization Principle (SRM). The goal of SVM is to determine the optimal hyperplane of feature space partition and the core idea of SVM is to maximize the classification of marginal. Structured output SVM tracking is used to directly estimate the change in target position between frames by learning a predictive function $f: X \rightarrow Y$, so the output space is a combination of the deformation of Y, instead of traditional binary labels -1 and +1. In this method, the sample (x, y) as a whole, y is the change function of target optimal position, the target position at the time of *t* is $P_t = P_{t-1}{}^{\circ} y_t$. So the framework used to learn a predictive function f can be obtained by introducing the discriminant function $F: \mathrm{X} \times \mathrm{Y} \rightarrow \mathrm{R}$. The result can be predicted according to (3).

$$y_t = f\left(x_t^{p_{t-1}}\right) = \underset{y \in Y}{\operatorname{argmax}} F\left(x_t^{p_{t-1}}, y\right) . \quad (3)$$

In order to update the predictive function online, we mark every new tracking position $\left(x_t^{p_t}, y^0\right)$. The discriminant function F measures the compatibility between (x, y) samples and chooses the best score in all samples. Equation (3) can be further expressed as $\mathrm{F}(x, y) = \langle w, \Phi(x, y) \rangle$ where $\theta(x, y)$ is a joint kernel projection. By minimizing the convex objective function (4), it can be learned in a new framework from a set of pairs $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.

$$\min_{w} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i \,. \tag{4}$$

$$\text{s.t. } \forall i: \xi_i \geq 0 \;\; \forall i, \forall y \neq y_i : \langle w, \delta\Phi_i(y) \rangle \;\geq\; \Delta(y_i, y) - \xi_i$$

$$\delta\Phi_i(y) = \Phi(x_i, y_i) - \Phi(x_i, y) \,.$$

The original Struck algorithm contains two sets of data in the support vectors machine (vector x). One is the features of the previous frame reserved support vectors, the other is the parameters that record the current support vector and their corresponding coefficients β. In the analysis of SVM algorithm, matrix M is calculated from vector x, matrix y and coefficient β.

$$\mathrm{M} \;= \sum_{i=1}^{m}\sum_{j=1}^{n} x_i \; x_j K\big(y_i, y_j, \beta\big)k \,. \tag{5}$$

Where $x_{i,j} \in R^2,\ \mathrm{i,j} = 1,2,\ldots,\mathrm{m,n}$. k is the adjustment parameter and K is kernel function. The calculation of matrix M occupies a large amount of computation time in CPU. To accelerate execution speed, the algorithm is computed in parallel, and each kernel thread in block (in GPU) only computes the row of matrix M, which means the calculation of initial matrix M is split into n subtasks and each subtask is executed asynchronously. The formula of each subtask is defined as follows.

$$\mathrm{M} \;= \sum_{i=1}^{m} {x_i}^2 \, K(y_i, y_i, \beta) \,. \tag{6}$$

This computational framework design is based on CUDA programming model and every kernel function runs a number of threads which are executed in parallel. In kernel programming, kernel thread only executes a simple cycle to improve the parallel procession of the algorithm.

4 Experimental Results

Hardware Environment: a standard PC equipped with 4G Memory, Dual-core Intel(R) Core(TM) i3-3220, 3.3GHz, GTX 750 TI graphics card with 640 CUDA cores, DJI Matrice 100. Software Environment: Windows 7, Microsoft Visual Studio 2010，CUDA 6.5, OpenCV2.4.10.

It is important to collect a representative dataset for a comprehensive and impartial performance evaluation of the tracking algorithm. The groundbreaking in this work is the VTB dataset presented in which contains 100 sequences from recent literatures, and it is the most influential visual target tracking dataset. Another influential visual target

tracking dataset is the VOT dataset. In this section we will use the VTB dataset with different frame sizes and video captured with UAV flying over traffic intersections to perform the tracking test. This part outlines the results of experiments which carried out during this research.

4.1 Evaluation of VTB Test Set

Object tracking experiments can be divided into two parts. In this part, five VTB image sequences are selected at random and manually marked for the ground truth which contains 2926 image frames. This experiment provides a comparison between the processing speed on GPU with CUDA and CPU for object tracking, and the resolution of picked image is identified below the figure. The record is measured once every 10 frames, 20 frames or 50 frames according to different sequence. Take the limitation of the length into consideration, we demonstrated the detailed results of the RedTeam public dataset to test the improved tracking algorithm, the other datasets only give the final statistical results.

We compare the processing speed of original CPU-only algorithm with the speed of GPU-accelerated code. The GPU-accelerated algorithm contains the image features calculation, dual SVM, learning/adaptation and multiple-kernel learning. As shown in Fig. 2, it provides details about the timing of processing the frames (the frame interval is 50 every record) of the proposed approach run on the video such as Fig. 3. The overall runtime of processing in VTB dataset are observed between the CPU and GPU versions. The results show that the speed of algorithm processes a video stream in GPU platform achieves a great effect than running on original CPU-only algorithm which is running about 2500-3000ms every 10 frames, and the speed running on CPU platform is unstable which changes intensely.

Fig. 4 and Tab. I show the results of performance comparison of tracking algorithm execution on CPU and GPU platform. In CarDark dataset, the speed boost obtained due to using a massive parallel CUDA programming model changes from 3.04 fps to 33.99 fps which achieved nearly 11 times speedup per second in comparison with CPU implementation. From the Fig. 4 we can see that the average FPS from the CPU implementation is below four frames per second which means the overall execution time will become greater as time goes on.

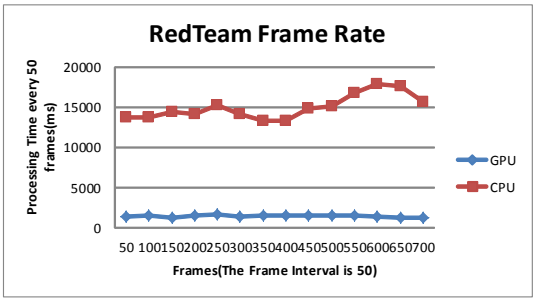


Fig. 2. Comparison of Calculation Performance on CPU and GPU in RedTeam Dataset at 352×340 Resolution.

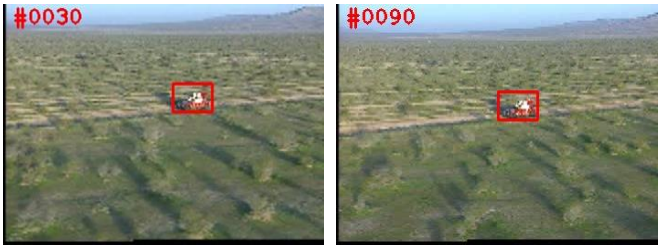


Fig. 3. RedTeam Dataset Tracking.

Table 1. The Statistics Table for Acceleration Factor.

Dataset	FPS in CPU	FPS in GPU	Acceleration Factor
Bike	3.09	21.26	5.88
CarDark	3.04	33.99	10.18
Couple	3.39	33.41	8.86
RedTeam	3.33	35.1	9.54
Surfer	3.22	31.03	8.64

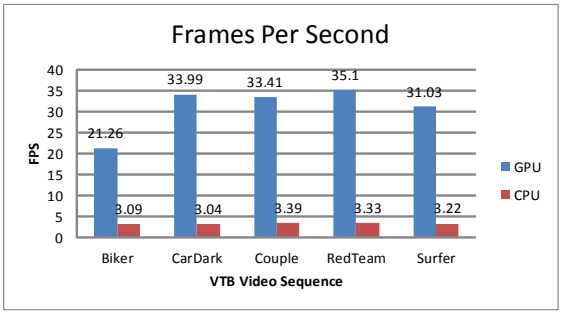


Fig. 4. FPS Test of RedTeam.

Next, the tracking performance will be valued on RedTeam dataset (as shown in Fig. 3) using two popular evaluation criteria: center location error (CLE) and overlap ratio (VOR). In order to reduce the accident, each video sequence is processed twice and picked the RedTeam sequence to show the performance in Fig. 5 and Fig. 6. The improved ratio between the original algorithm on CPU and improved algorithm on GPU in terms of center location error and overlap rate can be seen from the figure. In Fig. 5, the average CLE of original code is about 8.5 but the average CLE on GPU is only 2.4 which means the better capability of vehicle tracking in our method. In Fig. 6, the improved Struck algorithm is 31.63% higher than original Struck algorithm in average

tracking overlap ratio, and the average high score of VOR shows a good adaptability to complex environment.

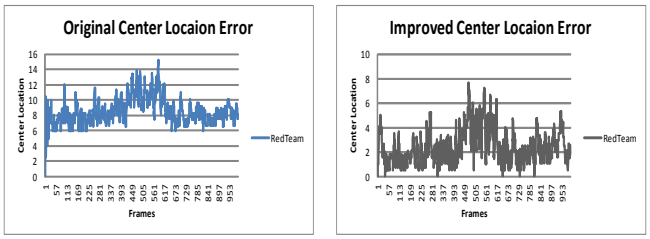


Fig. 5. Quantitative Comparison of CPU and GPU Versions in CLE on the RedTeam Video.

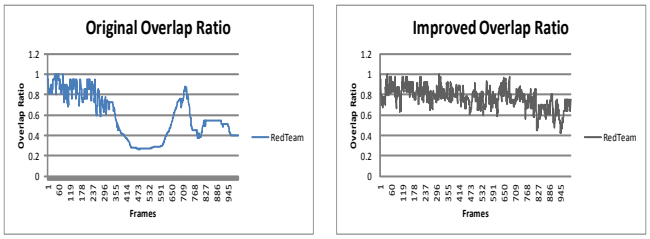


Fig. 6. Quantitative Comparison of CPU and GPU Versions in VOR on the RedTeam Video.

4.2 Evaluation of UAV Test Set

In this section, three scenes are captured by the DJI X3 camera with UAV in Dalian, Chinese and handled as follows: scene one contains the UAV shooting multi-targets. Scene two consists of the UAV overlooking a white car on a campus road and scene three includes the UAV overlooking a truck from a moving perspective. The purpose of the experiments was to compare the performance of the improved algorithm on CPU and GPU platforms on the same PC. The precision rate is not considered in this experiment, the time to process the video is what we care about.

In order to get a reasonable comparison, images with different resolutions are used to test. Fig. 8 shows the comparison of FPS (frames per second) running on CPU and GPU with different resolutions. From Fig. 8 we can see that the implementation of GPU algorithm has achieved a very significant acceleration effect, lower resolution of image means faster processing speed on GPU. The FPS of original algorithm on CUP platforms improves little despite the resolution of image varies greatly. The average speed of tracker using CUDA parallel optimization is about 4.5-5 times FPS in comparison with CPU implementation. Fig. 7 is the tracking performance in different UAV testing sequences.



Fig. 7. Vehicle Tracking in UAV Videos.

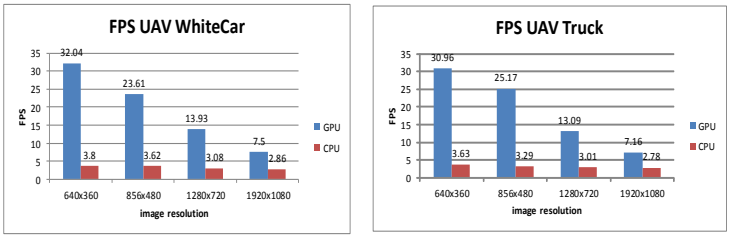


Fig. 8. FPS Test at Different Resolutions of White Car and Truck Dataset.

5 Conclusions

In this paper, structured output support vector machine and haar-like features has been used to improve the original Struck tracking method and accelerate the overall runtime with the implementation of improved algorithm on CUDA. The algorithm processes a video sequence with a frame rate close to 31 frames per second, and it is significant excellent than CPU. This method has been tested on UAV videos captured by DJI M100 UAV, and the experimental results show that the proposed algorithm is feasible to track and identify moving vehicles. In the future, the program should be optimized to better integrate into the CUDA programming architecture to achieve a better performance.

Acknowledgments. The work was supported by the National Natural Science Foundation of China (No. 61001158, 61272369 and 61370070), Liaoning Provincial Natural Science Foundation of China (Grant No. 2014025003), Scientific Research Fund of Liaoning Provincial Education Department (Grant No. L2012270), Specialized Research Fund for the Doctoral Program of Higher Education (SRFDP no 200801511027), and Fundamental Research Funds for the Central Universities (Grant No. 3132015084).

References

1. Callam, A.: Drone wars: Armed unmanned aerial vehicles. International Affairs Review, 18. (2015)
2. Springer, P. J.: Military robots and drones: a reference handbook. ABC-CLIO, (2013)
3. Yu, Q., Medioni, G.: Motion pattern interpretation and detection for tracking moving vehicles in airborne video. Computer Vision and Pattern Recognition, pp. 2671–2678. (2009)
4. Zhang, C., Kovacs, JM.: The application of small unmanned aerial systems for precision agriculture: a review. Computer Vision and Pattern Recognition, pp. 693–712. (2012)
5. Fu, C., Duan, R., Dogan, K., Erdal, K.: Onboard robust visual tracking for UAVs using a reliable global-local object model. Sensors, vol. 1406, (2016)
6. Pei, W., Zhu, Y., Zuo, X.: A multi-scale vehicle tracking algorithm based on Structured Output SVM. International Congress on Image & Signal Processing. pp. 1--6. (2015)
7. Hare, S., Saffari, A., Torr, P.: Struck: Structured output tracking with kernels. International Conference on Computer Vision. pp. 263--270. (2011)
8. Máttyus, G., Benedek, C., Szirányi, T.: Multi Target Tracking on Aerial Videos. Isprs Workshop on Modeling of Optical Airborne & Space Borne Sensors. pp. 1--7. (2010)
9. Ramakrishnan, V., Kethsy, A., Devishree, J.: A Survey on Vehicle Detection Techniques in Aerial Surveillance. International Journal of Computer Applications. pp. 43--47. (2012)
10. Reilly, V., Solmaz, B., Shah, M.: Shadow Casting Out Of Plane (SCOOP) Candidates for Human and Vehicle Detection in Aerial Imagery. International Journal of Computer Vision. pp. 350--366. (2013)
11. Uzkent, B., Hoffman, M.J., Vodacek, A.: Real-Time Vehicle Tracking in Aerial Video Using Hyperspectral Features. Computer Vision & attern Recognition Workshops. pp. 36--44. (2016)
12. Chen, B.J., Medioni, G.: Motion propagation detection association for multi-target tracking in wide area aerial surveillance. Advanced Video and Signal Based Surveillance. pp. 1--6. (2015)
13. Benatia, A., Ji, W., Wang, Y., Shi, F.: Sparse Matrix Format Selection with Multiclass SVM for SpMV on GPU. International Conference on Parallel Processing. pp. 496--505. (2016)
14. Zorski, W., Sklodowski, P.: Object tracking and recognition using massively parallel processing with CUDA. International Conference on Methods & Models in Automation & Robotics. pp. 977--982. (2015)
15. Rao, GM.: GPU Based Video Tracking System. IEEE Tenth International Conference on Semantic Computing. pp. 170--171. (2016)
16. Luebke, D.: Scalable parallel programming for high-performance scientific computing. IEEE International Symposium on Biomedical Imaging: from Nano to Macro. pp. 836--838. (2008)
17. Papageorgiou, C., Poggio, T.: A trainable system for object detection. International Journal of Computer Vision. pp. 15--33. (2000)