WEP Warwick Evans Publishing

International Journal of Mechanical and Electrical Engineering

ISSN: 3005-9615 (Print), ISSN: 3005-7132 (Online) | Volume 4, Number 3, Year 2024
DOI: https://doi.org/10.62051/ijmee.v4n3.08
Journal homepage: https://wepub.org/index.php/IJMEE/index

Fused High-Speed UAV Target Tracking Algorithm

Yu Zhang *, Mengqi Cui, Baokun Ji, Haojian Wang

School of Mechanical Engineering, Tianjin University of Technology and Education, Tianjin, China
* Corresponding Author: Yu Zhang

ABSTRACT

To achieve precise tracking and aiming in ground-based laser strikes against drones, this paper proposes a high-speed target tracking algorithm adapted for UAVs (Unmanned Aerial Vehicle) in aerospace backgrounds. The algorithm integrates the KCF algorithm with a custom simple template matching algorithm and optimizes the custom template matching method using NEON. On the Nvidia Orin NX, this approach significantly improves tracking real-time performance compared to using KCF alone. Applying this tracking strategy to a custom dataset and parts of the ANTI-UAV dataset yielded good tracking results.

KEYWORDS

KCF; Template Matching; NEON Optimization; High-Speed Tracking.

1. INTRODUCTION

In recent years, with the rapid advancement of technology, drones have seen increasingly widespread applications in both military and civilian fields[1, 2]. However, the frequent incidents involving drones pose a serious threat to national defense and social stability. Traditional defense measures have proven inadequate in dealing with drones, which has led to significant attention and in-depth research from major military powers on tactical laser weapon systems as a new type of defense measure[3, 4]. Tactical laser weapons have shown great potential in counter-drone operations, but their effective operation relies on precise and stable tracking technology[5, 6].

To achieve the interception and destruction of drones using tactical laser weapons, it is necessary to continuously and stably apply a specific power/energy density to a fixed position on the drone's body for several seconds[7]. To meet this objective, a refined three-stage aiming process is required: first, detecting and capturing the drone and guiding the tracking mount to bring the target into the coarse tracking field of view; then, the coarse tracking system precisely controls the tracking mount to ensure stable tracking of the drone within the fine tracking field of view, though the precision of coarse tracking alone is insufficient for precise strikes. Finally, the fine tracking system, through the control of a fast reaction mirror, finely corrects mechanical errors and tracking residuals to meet high-precision tracking requirements and locks onto a specific point on the target, ensuring the accuracy and effectiveness of the strike.

In the fine tracking stage, the camera not only needs to have a high frame acquisition rate, but its angle should also adjust in real-time with the drone's movement, imposing high real-time requirements on the image processing and recognition system. Moreover, fine tracking algorithms typically need to be deployed on embedded platforms, which often have limited resources such as processor performance and memory size. These constraints can severely impact the algorithm's running speed, further reducing its real-time performance. Therefore, when designing fine tracking

algorithms, it is essential to consider not only their accuracy and speed but also the characteristics of embedded platforms to achieve efficient operation and meet fine tracking requirements.

Considering that on embedded platforms, correlation filtering methods have high robustness but poor real-time performance, while local template matching methods have high real-time performance but poor robustness, this paper proposes a hybrid tracking method based on kernelized correlation filters and fast local template matching. This method ensures robust target tracking and accurate target following on embedded platforms while enhancing tracking real-time performance[8].

2. KERNELIZED CORRELATION FILTERS

KCF (Kernelized Correlation Filters) is a real-time object tracking algorithm based on correlation filtering, proposed by Joao F. Henriques et al. in 2014. The KCF algorithm integrates circular convolution, kernel tricks, and the Fast Fourier Transform (FFT) to achieve efficient target localization, particularly suited for handling high frame-rate video sequences. The core idea of KCF is to train a correlation filter such that it produces maximum response at the target location while minimizing responses elsewhere. This filter is trained using samples of the target from previous frames, employing a circular convolution form for learning. Circular convolution allows the filter to simulate an infinitely repeating template within a finite spatial scope, thereby simplifying computations and avoiding boundary effects [9].

3. ALGORITHM IMPROVEMENT

3.1. Design of the Targeting and Tracking Algorithm

To differentiate between the UAV and the sky background, we first apply a binarization process. Post binarization, a tracking algorithm that fuses the KCF (Kernelized Correlation Filters) algorithm with a custom-designed fast template matching algorithm is employed. To further enhance the processing speed on embedded devices, the binarization and custom template matching methods are optimized using NEON instructions. The flowchart of the algorithm is illustrated in Figure 1.

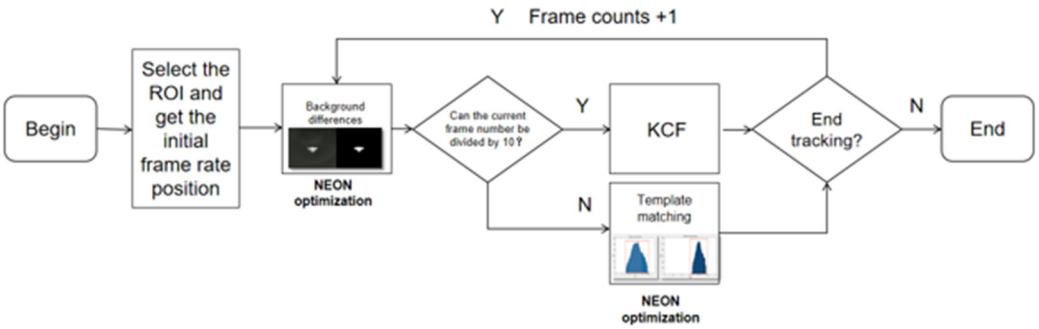


Figure 1. Flowchart of the Fused High-Speed UAV Target Tracking Algorithm

3.1.1. Background Binarization and NEON Optimization

In fine tracking scenarios, the image primarily consists of the target UAV and the sky background, with a significant contrast in pixel values between the two. This allows for effective differentiation through binarization. The binarized image provides clearer edge and feature recognition, facilitating subsequent image processing and analysis.

Choosing an appropriate binarization threshold for the background frame is crucial. An overly high threshold may misclassify the foreground target as background pixels, leading to missed detections, especially in situations where the foreground target appears dark or when the background noise is high. Conversely, a too-low threshold may incorrectly classify background pixels as foreground

targets, resulting in false detections when there are significant background variations. Thus, it is vital to select an optimal threshold.

This study employs Otsu's method to automatically determine the optimal threshold value. Prior to calculating this threshold, the grayscale histogram of the image is computed. Otsu's method then analyzes this histogram to identify the threshold that maximizes the inter-class variance between the segmented UAV and the sky background. To enhance the efficiency of the binarization process, the ARM architecture's SIMD (Single Instruction, Multiple Data) instruction set NEON is used to process multiple pixel values in parallel. Tests conducted on the ANTI-UAV dataset demonstrated effective differentiation between the UAV and the sky background. The binarization results are shown in Figure 2.

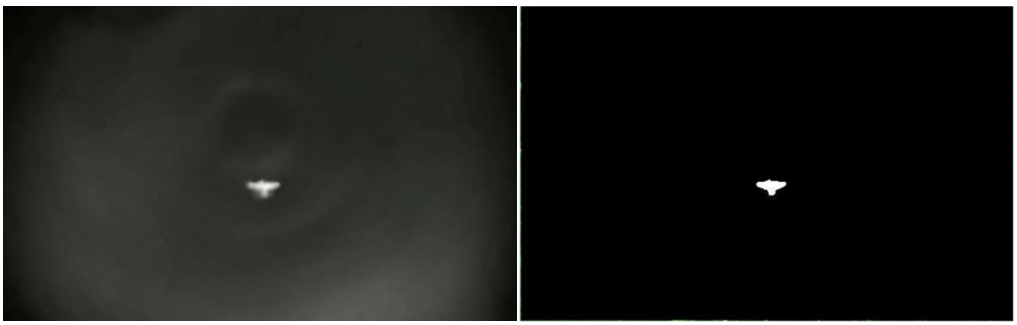


Figure 2. Otsu's Binarization Effect on a Sky Background

Once an appropriate threshold is identified, NEON optimization is employed to increase the efficiency of the binarization operation. The process involves:

Creating NEON vectors containing the threshold value and the maximum value (255).

Grouping pixels in sets of 16 and using the NEON instruction vld1q_u8 to load pixel data from memory into the vector in a single operation.

Executing the vcgeq_u8 instruction to compare the pixel vector with the threshold vector element-wise, generating a mask vector.

Using the vandq_u8 instruction to bitwise-AND the mask vector with the maximum value vector. This operation sets the pixel values that meet the threshold condition to 255, and the rest to 0.

Finally, employing the vst1q_u8 instruction to write the processed 16 binarized pixels back to memory in one operation.

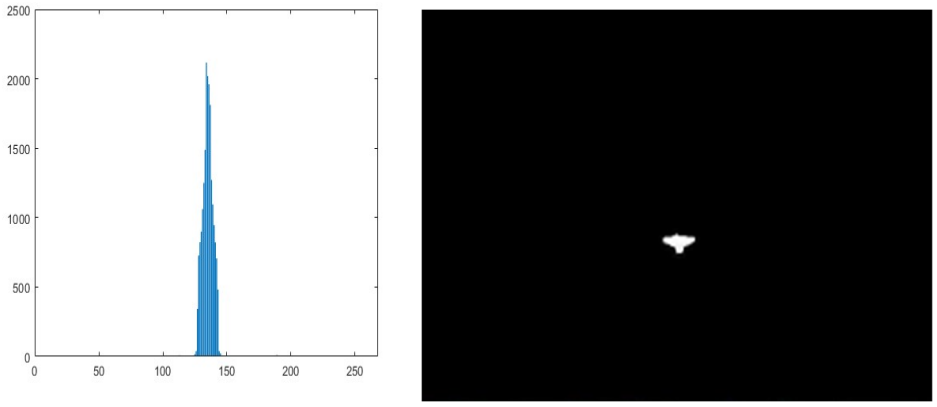
Through this method, NEON optimization enables parallel processing of 16 pixels at a time, reduces memory access times, and simplifies control flow, thereby significantly enhancing the execution speed of the binarization operation.

3.1.2. Custom Fast Template Matching and Its NEON Optimization

As previously mentioned, in the fine-tracking field of view, the image typically contains only the target and the sky background. Traditional template matching methods require collecting and matching the entire template pixel points, resulting in high computational costs and long matching times, which are challenging to meet the real-time demands of high-speed UAV tracking. To address this issue, this paper proposes a simplified and rapid template matching method accelerated by NEON, which can be integrated with Kernelized Correlation Filters (KCF) tracking algorithm to significantly enhance the overall system processing speed.

In this scenario, after appropriate binarization, the target (UAV) and the background (sky) can be distinctly separated in the binarized image. The pixel value distribution in the binarized image usually exhibits a unimodal distribution, as shown in Figure 3. Moreover, high-frame-rate cameras are

typically used in fine tracking. For the same actual target motion state, the higher the frame rate, the smaller the changes between consecutive frames of the target and environment. Thus, the UAV's position and attitude in the next frame will not change significantly from the previous frame, and the binarized image will continue to maintain a similar unimodal pixel value distribution. We can leverage this characteristic for matching after obtaining the binarized image.



(a) Binarized Image (b) Pixel Value Distribution

Figure 3. Pixel Value Distribution of Binarized Image

First, we calculate the sum of pixel values for each row and column in the binarized image within the region of interest (ROI) of the previous frame and store the results in two arrays. Next, we compute the similarity of all possible matching positions within the template region and the search area. By introducing a scoring mechanism, the algorithm evaluates the match quality of each potential matching position by calculating the sum of squared errors during the matching process. The closer the match, the higher the score. The score is derived by mapping the error within an acceptable maximum error range to ensure the scoring's effectiveness and sensitivity. Based on the scores, we determine whether the target is lost.

By further integrating the KCF tracking algorithm, we can continuously track the UAV's dynamic position changes rather than just performing static template matching. KCF tracking, due to its frequency domain processing, maintains high temporal efficiency and reliable tracking performance. This combined method not only accelerates the computation process but also maintains the algorithm's adaptability to fast-moving targets by continuously updating the tracking template. This design is suitable for application environments with high real-time response requirements, providing an efficient solution for UAV tracking, enabling fast and accurate tracking even under low target-background contrast conditions.

Since most image frames in the tracking process are handled by template matching, utilizing NEON's SIMD (Single Instruction, Multiple Data) instruction set allows us to vectorize the computation process, significantly improving template matching efficiency and, consequently, the overall tracking speed. The primary computational load affecting template matching speed lies in the template matching calculation. When using NEON to accelerate template matching, we first perform parallel counting operations on the pixel values in each row (or column) of the binarized image. Utilizing NEON's `vcntq_u8` instruction, we can process 16-pixel values at a time, computing the count of 1s in each value. This method considerably enhances processing speed compared to serial counting. Then, we use the `vaddvq_u8` instruction to sum the counted results, obtaining the total pixel value sum for the entire row or column.

Additionally, to further improve performance, we use NEON's load and store instructions (such as `vld1q_u8` and `vst1q_u8`) when processing each row or column's elements. These instructions efficiently load and store 16 bytes of data from memory, reducing memory access frequency and consequently lowering latency caused by memory access.

Following these optimizations, we conducted speed tests on the original normalized cross-correlation template matching, the custom fast template matching, and the NEON-accelerated custom fast template matching on the NVIDIA Jetson Orin NX. By running the matching methods on two 640×512 pixel images, we obtained the following results:

- The normalized template matching took 0.030823 seconds per frame, as shown in Figure 4.

- The custom fast template matching took 0.000336628 seconds per frame, as shown in Figure5.

- The NEON-optimized custom fast template matching took 0.000106985 seconds per frame, as shown in Figure 6.

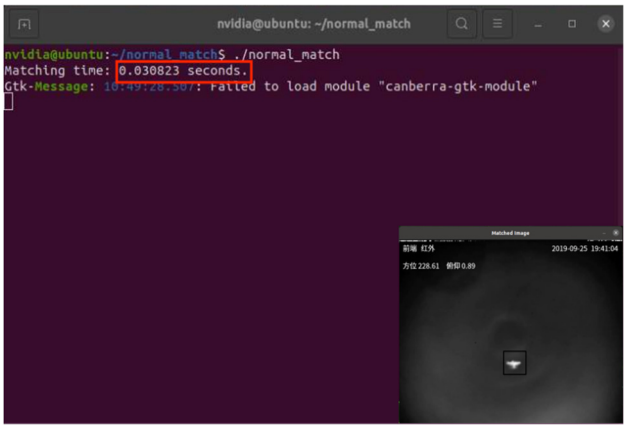


Figure 4. Time Consumption and Matching Results of Normalized Template Matching for 1 Frame

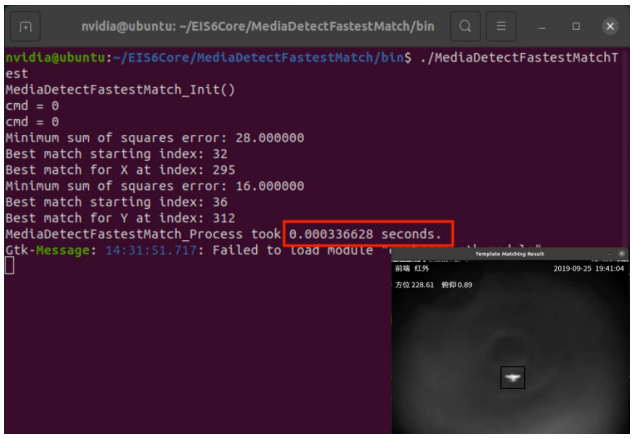


Figure 5. Time Consumption and Matching Results of Custom Fast Template Matching for 1 Frame

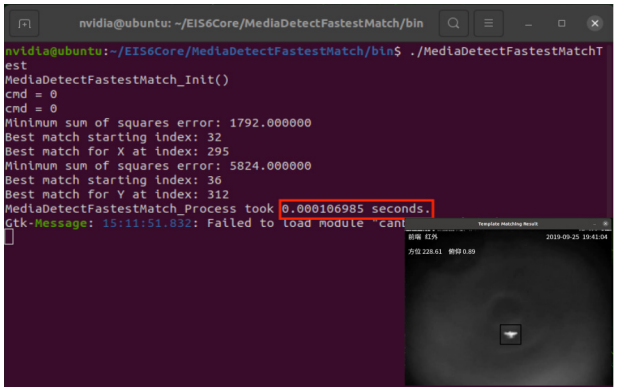


Figure 6. Time Consumption and Matching Results of Custom Fast Template Matching with NEON Optimization for 1 Frame

Based on the above results, Table1 is compiled, we observe that the custom fast template matching method proposed in this paper reduces the time to match one frame from the original 0.030823 seconds to 0.000336628 seconds, making it approximately 91.59 times faster than the normalized template matching method. Additionally, the NEON-optimized custom fast template matching processes one frame in only 0.000106985 seconds, which is about 3.145 times faster than the custom fast template matching method without NEON acceleration.

Table 1. Time Consumption of Different Template Matching Methods for 1 Frame

Matching Method	Matching Time
Normalized Template Matching	0.030823s
Custom Fast Template Matching	0.000336628s
Custom Fast Template Matching with NEON Optimization	0.000106985s

4. TRACKING EXPERIMENT WITH AIMING ALGORITHM

The tracking algorithm was deployed on the NVIDIA Jetson ORIN NX embedded AI computing platform for tracking experiments. The main parameters of the computing platform are shown in Table 2. The tracking algorithm was run on a custom UAV fine tracking dataset and a small portion of the ANTI-UAV dataset to observe its tracking performance. Some of the tracking result images are shown in Figure 7.

Table 2. Main Parameters of NVIDIA Jetson ORIN NX

Platform Model	Processor	GPU	Memory	Jetpack Version
NVIDIA Jetson ORIN NX	Cortex-A78AE	NVIDIA Ampere	16g	Jetpack5.1.1

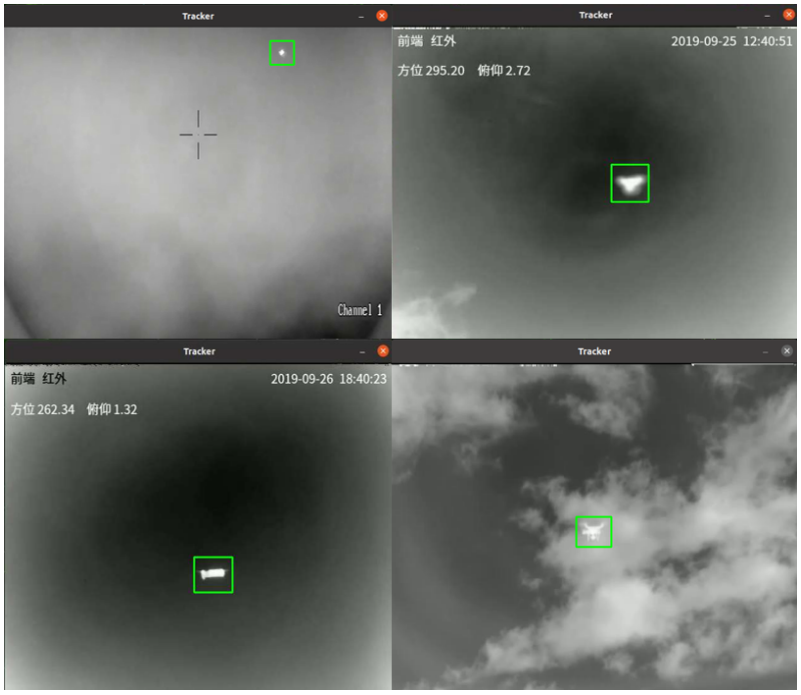


Figure 7. Partial Tracking Result Images

In the application scenario of this study, the UAV typically occupies a pixel size of 100x100 pixels. For targets of this size, tracking was performed on the embedded platform. Comprehensive tests on

the complete test set for both the KCF algorithm and the proposed algorithm yielded the results shown in Table 3. As shown in the table, this fuse tracking strategy can achieve an average tracking speed of around 250 FPS and good tracking accuracy on the test set. The processing speed is 2.8 times faster than the KCF algorithm, essentially meeting the tracking requirements for counter-UAV missions.

Table 3. Economic Data Statistics

Algorithm	Average Processing Speed (s/frame)	Frames/Second (fps)	Total Time (s)	Number of Frames
Ours	3.84647	259.979	20.244	5263
KCF	10.7861	92.7117	56.7674	5263

5. CONCLUSION

This paper proposes a high-speed target tracking algorithm for UAVs by integrating the Kernelized Correlation Filters (KCF) algorithm with a custom simple template matching algorithm. Additionally, it leverages the NEON instruction set to optimize the template matching method and background subtraction method, achieving efficient operation on the embedded platform Nvidia Jetson Orin NX. Experimental results show that this integrated tracking strategy significantly enhances the real-time performance of the tracking algorithm, addressing the speed insufficiency of traditional methods on embedded platforms. Specifically, this algorithm greatly improves the real-time performance of tracking on the Nvidia Orin NX compared to using only KCF. The NEON-optimized binarization and template matching techniques reduce the single-frame processing time from the original 0.030823 seconds to 0.000106985 seconds, achieving a speed increase of over 290 times. Furthermore, tests on the ANTI-UAV dataset and a custom UAV fine tracking dataset show that the algorithm can track targets with a template size of 100x100 pixels at an average speed of around 250 FPS while maintaining a tracking accuracy of over 80%, demonstrating excellent performance.

In summary, the tracking algorithm proposed in this paper effectively balances the robustness and real-time performance of target tracking algorithms on embedded platforms and shows strong adaptability in practical applications, meeting the high real-time requirements of precise UAV strike missions. Future research will further optimize the algorithm's performance and explore more application scenarios to enhance the actual effectiveness of tactical laser weapon systems in counter-UAV operations.

REFERENCES

[1] Comaniciu, D., Ramesh, V., & Meer, P. (2003). Kernel-based object tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(5), 564–577. https://doi.org/10.1109/TPAMI.2003.1195991.

[2] Kalal, Z., Mikolajczyk, K., & Matas, J. (2012). Tracking-Learning-Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(7), 1409–1422. https://doi.org/10.1109/TPAMI.2011.239.

[3] Hare, S., Golodetz, S., Saffari, A., Vineet, V., Cheng, M., Torresani, L., & Torr, P. H. S. (2016). Struck: Structured Output Tracking with Kernels. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(10), 2096–2109. https://doi.org/10.1109/TPAMI.2015.2509974.

[4] Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2015). High-Speed Tracking with Kernelized Correlation Filters. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(3), 583–596. https://doi. org/ 10. 1109/ TPAMI.2014.2345390.

[5] McErlean, M. (2006). An FPGA Implementation of Hierarchical Motion Estimation for Embedded Object Tracking. In 2006 IEEE International Symposium on Signal Processing and Information Technology (pp. 242–247). IEEE. https: // doi.org/10.1109/ISSPIT.2006.270805.

[6] Hsu, Y.-P., Miao, H.-C., & Tsai, C.-C. (2010). FPGA implementation of a real-time image tracking system. In Proceedings of SICE Annual Conference 2010 (pp. 2878–2884). IEEE.

[7] Elkhatib, L. N., Hussin, F. A., Xia, L., & Sebastian, P. (2012). An optimal design of moving objects tracking algorithm on FPGA. In 2012 4th International Conference on Intelligent and Advanced Systems (ICIAS2012) (pp. 745–749). IEEE. https://doi.org/10.1109/ICIAS.2012.6306112.

[8] Wong, S., & Collins, J. (2012). A proposed FPGA architecture for real-time object tracking using commodity sensors. In 2012 19th International Conference on Mechatronics and Machine Vision in Practice (M2VIP) (pp. 156–161). IEEE.

[9] Cho, J. U., Jin, S. H., Pham, X. D., & Jeon, J. W. (2006). Object Tracking Circuit using Particle Filter with Multiple Features. In 2006 SICE-ICASE International Joint Conference (pp. 1431–1436). IEEE. https://doi.org/ 10. 1109/ SICE. 2006.315755.