

DOI: 10.15514/ISPRAS-2024-36(3)-20



## An accurate real-time object tracking method for resource constrained devices

A. Sardaryan, ORCID: 0009-0002-0317-624X <armen.sardaryan@student.rau.am>  
V. Sahakyan, ORCID: 0000-0002-7552-4904 <vardan.sahakyan@student.rau.am>  
V. Melkonyan, ORCID: 0000-0002-3584-1294 <vahagn.melkonyan@student.rau.am>  
S. Sargsyan, ORCID: 0000-0002-8831-4965 <sevak.sargsyan@rau.am>

Russian-Armenian University,  
123, Hovsep Emin st., Yerevan, 0051, Armenia.

**Abstract.** This paper addresses the challenge of single-object tracking on resource-constrained devices, a critical aspect for applications like autonomous drones and robotics. We propose an efficient real-time tracking system that leverages the strengths of transformer-based neural networks in combination with correlation filters. Our research makes several key contributions: first, we conduct a comprehensive analysis of existing object tracking algorithms, identifying their advantages and limitations in resource-constrained environments. Second, we develop a novel hybrid tracking system that seamlessly integrates both neural networks and traditional correlation filters. This hybrid system is designed with a switching mechanism based on perceptual hashing, which allows it to alternate between fast but less accurate correlation filters and slower but more accurate neural network-based algorithms. To validate our approach, we implement and test the system on the Jetson Orin platform, which is representative of edge computing devices commonly used in real-world applications. Our experimental results demonstrate that the proposed system can achieve significant improvements in tracking speed while maintaining high accuracy, thereby making it a viable solution for real-time object tracking on devices with limited computational resources. This work paves the way for more advanced and efficient tracking systems in environments where computational power and energy are at a premium.

**Keywords:** single-object tracking; real-time computing; drone vision; visual transformers; correlation filters; perceptual hashing

**For citation:** Sardaryan A., Sahakyan V., Melkonyan V., Sargsyan S. An Accurate Real-Time Object Tracking Method for Resource-Constrained Devices, *Trudy ISP RAN/Proc. ISP RAS*, vol. 36, issue 3, 2024. pp. 283-294. DOI: 10.15514/ISPRAS-2024-36(3)-20.

**Acknowledgements.** This work was supported by the Science Committee of RA (Research projects № 23AA-1B007 and 23AA-1B005).

## Точный метод отслеживания объектов в реальном времени для устройств с ограниченными ресурсами

A. Сардарян, ORCID: 0009-0002-0317-624X <armen.sardaryan@student.rau.am>  
B. Саакян, ORCID: 0000-0002-7552-4904 <vardan.sahakyan@student.rau.am>  
B. Мелконян, ORCID: 0000-0002-3584-1294 <vahagn.melkonyan@student.rau.am>  
C. Саргсян, ORCID: 0000-0002-8831-4965 <sevak.sargsyan@rau.am>

Российско-Армянский университет,  
123, ул. Овсена Эмина, Ереван, 0051, Армения.

**Аннотация.** В данной статье рассматривается проблема отслеживания одиночных объектов на устройствах с ограниченными ресурсами, что является критически важным аспектом для таких приложений, как автономные беспилотники и робототехника. Мы предлагаем эффективную систему отслеживания в реальном времени, которая использует сильные стороны нейронных сетей на основе трансформеров в сочетании с корреляционными фильтрами. Наше исследование вносит несколько ключевых вкладов: во-первых, мы проводим всесторонний анализ существующих алгоритмов отслеживания объектов, выявляя их преимущества и проблемы в условиях ограниченных ресурсов. Во-вторых, мы разработали новую гибридную систему отслеживания, которая объединяет в себе как нейронные сети, так и традиционные корреляционные фильтры. Эта гибридная система разработана с механизмом переключения, основанным на перцептивном хешировании, что позволяет ей чередовать быстрые, но менее точные корреляционные фильтры с более медленными, но более точными алгоритмами на основе нейронных сетей. Для проверки нашего подхода мы реализовали и протестировали систему на платформе Jetson Orin, которая является репрезентативной для ограниченных в ресурсах вычислительных устройств, обычно используемых в настоящих приложениях. Результаты наших экспериментов показывают, что предложенная система может значительно повысить скорость отслеживания, сохраняя при этом высокую точность, что делает ее жизнеспособным решением для отслеживания объектов в реальном времени на устройствах с ограниченными вычислительными ресурсами. Эта работа открывает путь к созданию более совершенных и эффективных систем отслеживания в условиях, когда вычислительная мощность и энергия находятся на пределе.

**Ключевые слова:** отслеживание одиночного объекта; вычисления в реальном времени; зрение дрона; визуальные трансформеры; корреляционные фильтры; перцептивный хеш.

**Для цитирования:** Сардарян A., Саакян B., Мелконян B., Саргсян C. Точный метод отслеживания объектов в реальном времени для устройств с ограниченными ресурсами. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 283–294 (на английском языке). DOI: 10.15514/ISPRAS-2024-36(3)-20.

**Благодарности.** Работа выполнена при поддержке Комитета по науке Республики Армения (исследовательские проекты № 23AA-1B007 и 23AA-1B005).

### 1. Introduction

Single-object tracking is a critical task in computer vision, involving the continuous localization and tracking of a single target in a video, given only a bounding box in the first frame. This task becomes increasingly complex when computational resources are limited. This is often the case in real-world applications that require tracking algorithms to run on resource-constrained platforms. Most of the current tracking algorithms depend on pre-trained models with a large number of parameters. While these algorithms may achieve high performance on benchmark datasets on GPU, they are either not real-time or cannot operate effectively on resource-constrained boards.

The main goal of this research is to develop and implement an efficient real-time object tracking system and to integrate it on resource-constrained platforms. To achieve this goal, it is proposed to utilize the advantages of both types of algorithms: speed of correlation filters and accuracy of slow neural networks. For this purpose, a hybrid method with a switch mechanism between the two algorithms is proposed.

## 2. Related Works

This paper focuses on two main classes of single-object tracking algorithms: correlation filters and deep learning algorithms, particularly transformers. These approaches are chosen due to their proven performance [1].

### 2.1 Correlation Filters

Correlation filters have been widely used in object tracking due to their efficiency. Bolme et al. introduced the Minimum Output Sum of Squared Error (MOSSE) [2] filter, which achieved high speeds but struggled with target appearance changes and background interference. Subsequent advancements like Kernelized Correlation Filters (KCF) [3] by Henriques et al. improved accuracy by using circulant matrices and kernel tricks. Danelljan et al. further enhanced tracking accuracy with Discriminative Scale Space Tracking [4], addressing issues like occlusions and scale variations. Danelljan et al. proposed the C-COT [5] algorithm, integrating convolutional features and continuous convolution filters for improved tracking quality, although at the cost of speed. They later introduced ECO (Efficient Convolution Operators) [6] to address these speed issues, simplifying features and reducing template updates while maintaining high tracking quality. The CSRT [7] algorithm by LuNežić et al. introduced spatial and channel reliability maps, improving robustness against occlusions and clutter.

### 2.2 Transformers

Based on the success of Transformers in NLP tasks, their idea has been used in other tasks as well. In 2021, Dosovitskiy et al. published the first paper on visual transformers [8], paving the way for their use in computer vision tasks. Visual transformers have emerged as a powerful tool in object tracking. Wang et al. [9] applied transformers to tracking by designing CNN-Transformer hybrid models, which significantly improved robustness and accuracy. Yan et al.'s STARK [10] tracker utilized transformers for both spatial and temporal feature extraction, setting new benchmarks in tracking performance. Recent innovations like HCAT [11] by Chen et al., MixFormer [12] and its successor MixFormerV2 [13] by Cui et al. have optimized transformer architectures for tracking, achieving state-of-the-art performance while addressing speed limitations.

### 2.3 Hybrid algorithms

Some studies have tried to address the trade-off between speed and accuracy of tracking algorithms by combining heavily accurate and fast but not accurate trackers. Such hybrid algorithms are often used on embedded devices because of their high speed and quality.

In the paper [14] by H. Mao et al., the algorithm switches to an accurate and slow tracker (Siamese network) every 4 frames and uses a lightweight neural network the rest of the time. In this case, switching between neural networks is very frequent, which also affects the speed, but the quality of the underlying lightweight algorithm is very slow to reduce the switching frequency. Besides that, the speed of this lightweight algorithm that uses PatchNet architecture for template matching is much lower than correlation filters that use hand-crafted features.

In the paper [15] Ji Q. et al. also considered a hybrid algorithm for use on resource-constrained platforms. They also, like us, use the correlation filter (KCF) as a fast algorithm and improve its quality with an object detection algorithm, SSD [16]. They use perceptual hashing to detect failures in KCF tracking, and then perform (re)detection of the object using SSD. Besides that, the SSD works each N frame anyway. Thus, this paper describes a tracking-by-detection algorithm, that is, it tracks the object using KCF throughout the video, adjusting the bounding box using the detection algorithm when needed. Working aboard a drone involves a lot of situations in which the KCF will fail, thus detection and double triggering of the KCF will be very frequent, which may affect the

speed. The speed of the proposed tracker reached 36FPS on ZCU104 and the quality improved significantly with respect to simple KCF.

## 3. Proposed Method

In this research paper, we propose a novel hybrid tracking method with a switch mechanism between fast and accurate algorithms to achieve a trade-off between speed and quality of tracking. The proposed algorithm combines visual transformer and correlation filters as the best quality and fastest types of trackers respectively.

### 3.1 Perceptual Hashing

When to switch from a fast algorithm to an accurate one is determined using perceptual hashing [17]. This method creates a compact, unique fingerprint for each image, capturing essential visual features. The perceptual hashing algorithm converts an image to grayscale, resizes it to a fixed size (e.g., 8x8 or 16x16), and calculates the average pixel intensity. Pixels are labeled 1 or 0 based on whether they are brighter or darker than the average value, forming the hash code. This technique is particularly useful for applications such as content-based retrieval, copyright protection, and digital forensics. Perceptual hashing contains information about the content of the image, so the slightest change in the content will change the hash code.

### 3.2 Tracking Method

Perceptual hashing allows us to efficiently compare the contents of an image, which we use to determine when a fast-tracking system ‘starts to lose’ an object. We will refer to the rectangle around the tracked object in a frame as the bounding box, and the image inside this rectangle as the region of interest (ROI). We extract ROIs from every two consecutive frames and compare their hash codes obtained by perceptual hashing. We use Hamming distance, which counts the number of differing bit positions of the hash codes as follows:

$$H(a, b) = \sum_{i=1}^n 1(a_i \neq b_i), \quad (1)$$

Where  $a, b$  are two hash codes of the same size  $n$ ,  $1(a_i \neq b_i)$  is the indicator function, equal to 1 if  $a_i \neq b_i$ , and 0 otherwise. Only two small frame regions are compared per iteration, so there is no significant load on the speed of the tracker.

First, we run the correlation filter as the main tracker. We assume that this tracker ‘starts to lose’ an object when  $H(a, b)$  exceeds a given threshold. Once this happens, our system switches the Correlation Filter to a more accurate Transformer by passing it to the bounding box of the previous frame. The transformer takes one tracking step on the current frame, thereby correcting its bounding box. In the next frame, the Correlation Filter is reinitialized and the system again decides which tracker to continue tracking depending on the number  $H(a, b)$ .

Note that before deciding which tracker to use to make a tracking step, we do not yet have a bounding box for the current frame. Regions of interest are extracted by overlaying the same bounding box (of the previous frame) on the previous and current frames. A large value of  $H(a, b)$  means that the object has significantly changed its position between neighboring frames. The correlation filter is unable to track such a change, which is why we switch to the more powerful Transformer at such moments.

Algorithm 1 shows the pseudocode of one step of the proposed hybrid tracking method. Threshold is a hyperparameter, which is tuned during the experiments.

#### Algorithm 1. One step by hybrid tracking method

**Input:** *current frame, previous frame, initialized Transformer and Correlation Filter, previous bounding box, threshold*  
**Output:** current bounding box

- 1: **if** previous tracking step was done by *Transformer*:
- 2:     **reinitialize** *Correlation Filter* with *previous bounding box*
- 3:     **extract** ROIs from *current* and *previous* frames using *previous bounding box*
- 4:     **compute** hash codes for the extracted ROIs
- 5:     **calculate** differing bits between the two hash codes
- 6:     **if** differing bits > *threshold*:
- 7:         **get** current bounding box by *Transformer*
- 8:     **else**:
- 9:         **get** current bounding box by *Correlation Filter*
- 10:     previous bounding box ← current bounding box

## 4. Experiments and Results

### 4.1 Experimental setup

The first part of the experiments is the selection of basic algorithms: one correlation filter and one transformer. The comparative analysis was performed on a personal computer with an 11th generation Intel® Core™ i7-11700 @ 2.50GH x86-64 CPU and an NVIDIA RTX 3060 GPU. Correlation filters were run on the CPU and transformers on the GPU. The following environment was installed on the PC to run the algorithms: Ubuntu 20.04, Python 3.8.10, gcc 9.4.0, Cmake 3.16.3, CUDA 12.2, TensorRT 8.6.0, Onnx 1.16.0, OpenCV 4.9.0.

The second part is experimenting with our hybrid method based on the two selected algorithms. These were performed both on a PC and on a resource-constrained device that can be used on board a UAV: Jetson Orin. The following environment was installed on Jetson Orin: Ubuntu 22.04, Python 3.10.12, gcc 11.4.0, Cmake 3.22.1, CUDA 12.2, TensorRT 8.6.2, Onnx 1.16.0, OpenCV 4.5.4.

### 4.2 Datasets

We have used the UAV123 [18] and VisDrone-SOT [19] datasets for algorithms testing. These datasets were chosen because they contain videos with relatively quick changes in perspective and lighting and sudden object movements. The videos are like this because they are taken from a UAV. Finding a balance between speed and quality of trackers is a critical task precisely in the context of the UAV onboard algorithms.

UAV123 dataset consists of 123 videos with a resolution of 1280x720, totaling over 110,000 frames. 103 videos were captured with a DJI drone from 5-25 meters at 30-96 fps, 12 videos were from a small low-cost drone without stabilization, and 8 were generated in Unreal Engine (around 30 fps). Each frame's annotation is  $[x, y, w, h]$ , where  $(x, y)$  is the top-left coordinate of the bounding box,  $w$  and  $h$  are its width and height. During full occlusion, NaN is recorded for each position  $(x, y, w, h)$ . The VisDrone-SOT dataset includes 167 videos. These videos have different resolutions, totaling over 188,000 frames, and are divided into train (86 videos), validation (11 videos), and test sets (60 videos). For our comparison, we have used all videos. The annotation for each frame is  $[x, y, w, h]$ , but unlike UAV123, occlusions are recorded with the inferred bounding box instead of NaN.

### 4.3 Metrics

Metrics from the last VOTS-2023 challenge [20], namely, Accuracy (A), Robustness (R), Not-Reported Error (NRE), Drift-Rate Error (DRE), and Absence-Detection Quality (ADQ) were

selected to evaluate the tracking quality. All of them are based on 5 main tracking scenarios, which are described in Fig. 1.

Accuracy (A) is defined as the ratio of correctly detected objects to all situations where the tracker detects something. In eq. (2), it is represented as:

$$A = \frac{N_{sc1}}{N_{sc1} + N_{sc2} + N_{sc4}}, \quad (2)$$

where  $N_{sc1}$  is the number of correct detections,  $N_{sc2}$  is the number of drifted situations, and  $N_{sc4}$  is the number of false detections.

Hereinafter, a correct detection is defined as one in which the Intersection of Union (IoU) is greater than the threshold that we set ourselves. IoU is calculated as the ratio of the overlap area of the ground-truth box and tracker's bounding box to their union area.

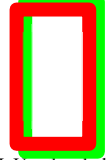
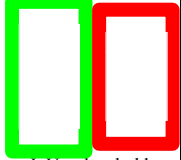
| Scenario 1. Success                                                                 | Scenario 2. Fail                                                                    | Scenario 3. Fail                                                                    | Scenario 4. Fail                                                                    | Scenario 5. Success |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|---------------------|
|  |  |  |  |                     |

Fig. 1. Possible scenarios during single object tracking. The red rectangle is the ground-truth box of the object, green is the bounding box of the tracker. 1 and 5 scenarios are successful, but 2-4 are not.

Robustness (R) is the ratio of correctly detected objects to all situations where an object is present. In eq. (3), it is

$$R = \frac{N_{sc1}}{N_{sc1} + N_{sc2} + N_{sc3}}, \quad (3)$$

where  $N_{sc3}$  is the number of missed detections.

Robustness together with Drift-Rate Error (DRE) and Not-Reported Error (NRE) give a total of one. DRE measures false (drifted when  $\text{IoU} < \text{threshold}$ ) detections when an object is present, while NRE measures missed detections when an object is present. They are given by:

$$DRE = \frac{N_{sc2}}{N_{sc1} + N_{sc2} + N_{sc3}}, \quad (4)$$

$$NRE = \frac{N_{sc3}}{N_{sc1} + N_{sc2} + N_{sc3}}. \quad (5)$$

Absence-Detection Quality (ADQ) measures how well the tracker detects the absence of an object. It is defined as:

$$ADQ = \frac{N_{sc5}}{N_{sc4} + N_{sc5}}, \quad (6)$$

where  $N_{sc5}$  is the number of correct absence detections.

For the above metrics, the IoU-threshold was set at 0.5 in all of our experiments. We also calculate the Area Under the Curve (AUC) for both Accuracy (AUC-A) and Robustness (AUC-R), which indicates the tracker's quality across different IoU-thresholds (ranging from 0 to 1 in steps of 0.01). The last but not least metric is frames per second (FPS), which indicates the speed of the tracker.

#### 4.4 Selection of Basic Algorithms

In this study, we conducted a two-stage comparison of algorithms to identify the most effective options. Initially, we evaluated various correlation filter-based methods to determine the optimal approach among them. Subsequently, we assessed transformer-based methods, comparing their performance against each other. The analysis in this section was conducted on a personal computer. The correlation filters MOSSE, KCF, CSRT, DSST, and ECO were selected for comparative analysis. The standard trackers MOSSE, KCF, and CSRT were taken from OpenCV. KCF was also implemented from scratch based on source paper and repository, as were DSST and ECO. The implementation of all correlation filters was written in C++ and built using Cmake.

The comparison of the correlation filters is shown in Table 1. ADQ metric is not considered on the VisDrone-SOT dataset due to the specificity of its annotation (see B). As you can see the two trackers with the highest FPS have very low-quality scores. The best option in terms of speed and quality (highest accuracy) is KCF implemented by us, so we will choose it for further experiments. The official HCAT implementation [21], the MixFormerV2 implementation with TensorRT and CUDA support [22], and the ViT Tracker implementation from the OpenCV\_zoo repository [23] were considered to select the transformer-based algorithm. NRE and ADQ metrics are not considered because of the specificity of transformers that localize the box for an object in each frame, even if the object does not exist there. The comparison of transformers is shown in Table 2. Based on analyzing the quality and speed of the algorithms on datasets, we chose the state-of-the-art algorithm MixFormerV2 for our further experiments.

Table 1. Comparison of correlation filters on UAV123 and VisDrone-SOT datasets (PC).

|                | UAV123 |       |           |           |         |         |         |       | VisDrone-SOT |       |           |           |         |         |      |
|----------------|--------|-------|-----------|-----------|---------|---------|---------|-------|--------------|-------|-----------|-----------|---------|---------|------|
|                | A (%)  | R (%) | AUC-A (%) | AUC-R (%) | ADQ (%) | NRE (%) | DRE (%) | FPS   | A (%)        | R (%) | AUC-A (%) | AUC-R (%) | NRE (%) | DRE (%) | FPS  |
| MOSSE (OpenCV) | 20     | 10.8  | 18.5      | 11.2      | 19.2    | 65.1    | 24.1    | 14750 | 29.8         | 23.2  | 27.5      | 21.9      | 46.1    | 30.7    | 9601 |
| KCF (OpenCV)   | 45.8   | 15.8  | 40.9      | 14.9      | 24.9    | 71.2    | 13      | 1696  | 60.9         | 25.9  | 52.3      | 22.7      | 56.7    | 17.4    | 916  |
| CSRT (OpenCV)  | 47     | 41.8  | 42.6      | 37.6      | 9.9     | 13.4    | 44.9    | 174   | 61.5         | 60.2  | 51.2      | 50.3      | 4       | 35.8    | 92   |
| KCF            | 54.1   | 42.4  | 43.8      | 34.3      | 12.3    | 26.4    | 31.2    | 478   | 60.2         | 57.5  | 48.5      | 46.2      | 7.8     | 34.7    | 450  |
| DSST           | 45.7   | 30    | 38.9      | 25.2      | 19.2    | 47.5    | 22.5    | 406   | 54.1         | 46.3  | 43        | 366       | 22.3    | 31.4    | 270  |
| ECO            | 43.3   | 43.4  | 35.4      | 35.4      | 3       | 6       | 50.6    | 151   | 57.9         | 57.8  | 46.5      | 46.4      | 0.5     | 41.7    | 128  |

Table 2. Comparison of transformer-based trackers on UAV123 and VisDrone-SOT datasets (PC).

|                      | UAV123 |       |           |           |         |     | VisDrone-SOT |       |           |           |         |     |
|----------------------|--------|-------|-----------|-----------|---------|-----|--------------|-------|-----------|-----------|---------|-----|
|                      | A (%)  | R (%) | AUC-A (%) | AUC-R (%) | DRE (%) | FPS | A (%)        | R (%) | AUC-A (%) | AUC-R (%) | DRE (%) | FPS |
| ViT Tracker (OpenCV) | 64.6   | 65.1  | 52        | 52.5      | 34.9    | 181 | 65.7         | 65.8  | 50.4      | 50.4      | 34.2    | 172 |
| HCAT                 | 75.4   | 76.2  | 60.8      | 61.6      | 23.8    | 140 | 74.8         | 74.8  | 58.6      | 58.6      | 25.2    | 133 |
| MixFormerV2 (TRT)    | 79.5   | 80.6  | 64        | 65        | 19.4    | 270 | 71.6         | 71.6  | 56.5      | 56.6      | 28.3    | 262 |

#### 4.5 Switching threshold

The tunable parameter of our hybrid method is the threshold, which the frequency of switching from KCF to MixFormerV2-S depends on. The window size for perceptual hashing is 8x8, hence the maximum possible number of bits in which a difference is possible is 64. We analyzed the

dependence of accuracy, FPS and percentage of MixFormerV2 responses on the threshold on the random 30 videos from both datasets (15 of each). These plots are shown in Fig. 2. The percentage of MixFormerV2 responses is shown by an additional axis within the plot.

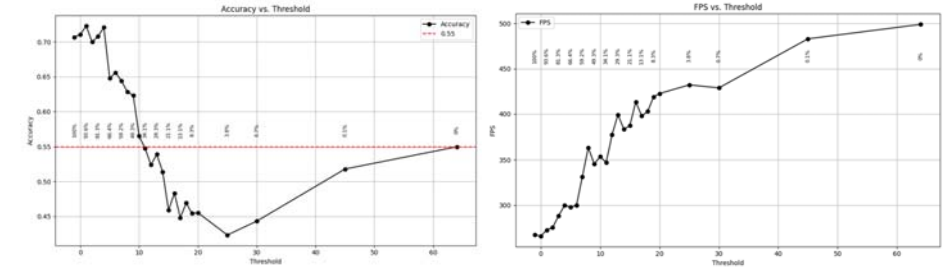


Fig. 2. Dependence of accuracy (left) and FPS (right) on threshold. The additional axis in the middle indicates the corresponding percentage of MixFormerV2 triggering.

As can be seen in Fig. 2, tracking accuracy drops even below KCF for threshold values greater than 11. This may be because, at high thresholds, switching occurs in situations where the object has already been lost, and MixFormerV2 tracks the wrong object, thereby passing a deliberately erroneous bounding box to KCF. Therefore, although the speed (FPS) increases as the threshold increases, values greater than 11 do not make sense to consider on these datasets. We chose thresholds 5 and 10 as reference points for full experiments on both datasets. MixFormerV2 and KCF (which correspond to thresholds -1 and 64, respectively) were also used for comparison.

#### 4.6 Experiments

##### 4.6.1 On a personal computer with GPU

Table 3 shows the comparison for all metrics except NRE and ADQ on both datasets in their entirety. The first column shows the average number of MixFormerV2 responses per algorithm.

At threshold equals 5, the number of MixFormerV2 responses on UAV123 is 68%, while on VisDrone-SOT it is 55%. The difference is explained by the video quality: the VisDrone-SOT's average resolution is much higher. Compared to MixFormerV2, accuracy drops by 1% and 2%, while FPS increases by 26% and 36% (respectively for UAV123 and VisDrone-SOT).

At threshold equals 10, the number of MixFormerV2 responses on UAV123 is 51%, while on VisDrone-SOT it is 30%. Compared to MixFormerV2, accuracy drops by 8% and 11%, while FPS increases by 36% and 59% (respectively for UAV123 and VisDrone-SOT). In terms of accuracy on the VisDrone-SOT dataset, this algorithm equaled KCF (60.2% for both), although the FPS is slightly lower. However, on UAV123 it still showed much better results than KCF (71% vs. 54.1%). MixFormerV2 performs worse on higher-resolution videos with small targets on them when in contrast KCF performs better. This is why the quality difference between KCF and MixFormerV2 on the VisDrone-SOT dataset is not that big. However, the results with threshold 5 are still quite good.

Table 3. Comparison of Hybrid tracking method with different thresholds  $T=5$  and  $T=10$  with MixFormerV2 and KCF on UAV123 and VisDrone-SOT datasets (PC).

|                   | UAV123  |       |       |           |           |         |     | VisDrone-SOT |       |       |           |           |         |     |
|-------------------|---------|-------|-------|-----------|-----------|---------|-----|--------------|-------|-------|-----------|-----------|---------|-----|
|                   | % of MF | A (%) | R (%) | AUC-A (%) | AUC-R (%) | DRE (%) | FPS | % of MF      | A (%) | R (%) | AUC-A (%) | AUC-R (%) | DRE (%) | FPS |
| KCF               | 0       | 54.1  | 42.4  | 43.8      | 34.3      | 31.2    | 478 | 0            | 60.2  | 57.5  | 48.5      | 46.2      | 34.7    | 450 |
| MixFormerV2 (TRT) | 100     | 79.5  | 80.6  | 64        | 65        | 19.4    | 270 | 100          | 71.6  | 71.6  | 56.5      | 56.6      | 28.3    | 262 |

|               |    |      |      |      |      |      |     |    |      |      |      |      |      |     |
|---------------|----|------|------|------|------|------|-----|----|------|------|------|------|------|-----|
| KCF-MF (T=5)  | 68 | 78.2 | 79.3 | 62.6 | 63.4 | 20.7 | 342 | 55 | 69.2 | 69.2 | 54.6 | 54.7 | 30.7 | 357 |
| KCF-MF (T=10) | 51 | 71   | 72   | 56   | 57   | 28   | 366 | 30 | 60.2 | 60.2 | 48   | 48.1 | 39.8 | 417 |

#### 4.6.2 On Jetson Orin

Similar experiments were conducted on a resource-constrained device, namely the Jetson Orin. In addition, we made a comparison with another transformer-based algorithm HCAT. Table 4 shows the comparison of selected algorithms on UAV123 and the VisDrone-SOT dataset on Jetson Orin.

Table 4. Comparison of Hybrid tracking method with different thresholds  $T=5$  and  $T=10$  with MixFormerV2, KCF and HCAT on UAV123 and VisDrone-SOT datasets (on Jetson Orin).

|                   | UAV123  |       |       |           |           |         |     | VisDrone-SOT |       |       |           |           |         |     |
|-------------------|---------|-------|-------|-----------|-----------|---------|-----|--------------|-------|-------|-----------|-----------|---------|-----|
|                   | % of MF | A (%) | R (%) | AUC-A (%) | AUC-R (%) | DRE (%) | FPS | % of MF      | A (%) | R (%) | AUC-A (%) | AUC-R (%) | DRE (%) | FPS |
| KCF               | 0       | 53.7  | 41.8  | 43.8      | 34.1      | 58.1    | 174 | 0            | 60.2  | 57.5  | 48.4      | 46.2      | 42.5    | 153 |
| MixFormerV2 (TRT) | 100     | 79.7  | 80.7  | 64.1      | 65        | 19.3    | 34  | 100          | 68.7  | 69.6  | 54.8      | 55.5      | 30.4    | 31  |
| KCF-MF (T=5)      | 73.3    | 76.4  | 77.2  | 61.3      | 62        | 22.8    | 54  | 62.2         | 66.4  | 66.3  | 52.3      | 52.4      | 33.7    | 68  |
| KCF-MF (T=10)     | 48      | 69.5  | 69.3  | 55        | 55        | 30.6    | 82  | 30.43        | 59.9  | 58.9  | 48.1      | 47.4      | 41.1    | 113 |
| HCAT              | -       | 75.2  | 76.1  | 60.7      | 61.4      | 23.9    | 33  | 0            | 74.6  | 74.6  | 58.3      | 58.4      | 25.4    | 33  |

As we can see, on the UAV123 dataset, our hybrid algorithm with threshold 5 gives a 59% speedup, while accuracy and robustness drop by only 3% compared to MixFormerV2. If we compare with HCAT, the accuracy and robustness of our algorithm is 1% higher each while the FPS is 64% higher. The accuracy and robustness of the algorithm with threshold 10 drop by 10% and 11% compared to MixFormerV2, but the speed increases by 141 %. Relative to HCAT, accuracy is 6 % worse, robustness is 7 % worse, but speed is 148 % higher.

Analyses of the VisDrone-SOT dataset showed different results. Possible reasons for this were explained above. At the same time, the HCAT quality did not change due to the high resolution of the video. As we can see, MixFormerV2 is 6% worse in accuracy, 5% worse in robustness, and 6% worse in FPS than HCAT. However, if we compare our hybrid algorithms with MixFormerV2, we again see that with accuracy and robustness dropping by only 2% and 3%, FPS rises by 119% (for the algorithm with threshold 5).

In Fig. 3 are plots of ROC curves showing the change of tracking accuracy and robustness depending on the IoU threshold. We can conclude that the quality of our algorithm with threshold 5 is better than HCAT (on UAV123) and only slightly inferior to MixFormerV2 (on both). With this quality, the 54FPS (UAV123) / 68FPS (VisDrone-SOT) rate on Jetson Orin looks attractive (compared to the 31-34FPS of MixFormerV2 and HCAT).

The proposed hybrid method allows us to adjust the desired ratio of quality and speed of tracking by setting the threshold hyperparameter. The algorithm itself decides at what moments to switch, and the frequency of switching depends on the video quality and the presence of abrupt scene changes. By increasing the threshold, we decrease the number of switches and thus increase the speed of tracking. At the same time, as can be seen from the experiments, as the threshold increases, the accuracy drops much slower than the FPS increases.

#### 5. Conclusion

In this paper, we proposed a hybrid object tracking method that combines correlation filter-based algorithms with transformers and determines the moment to switch between them using perceptual hashing. This technique achieves the necessary balance between speed and tracking accuracy. The proposed algorithm can play a crucial role in running on resource-constrained computers that are used on board UAVs or other robots. Experiments have shown that the proposed algorithm can indeed significantly improve the tracking speed relative to current state-of-the-art models without

significant loss of quality. On the UAV123 dataset, our hybrid algorithm showed a 59% speedup with only a 3% decrease in accuracy and robustness compared to MixFormerV2 (on Jetson Orin).

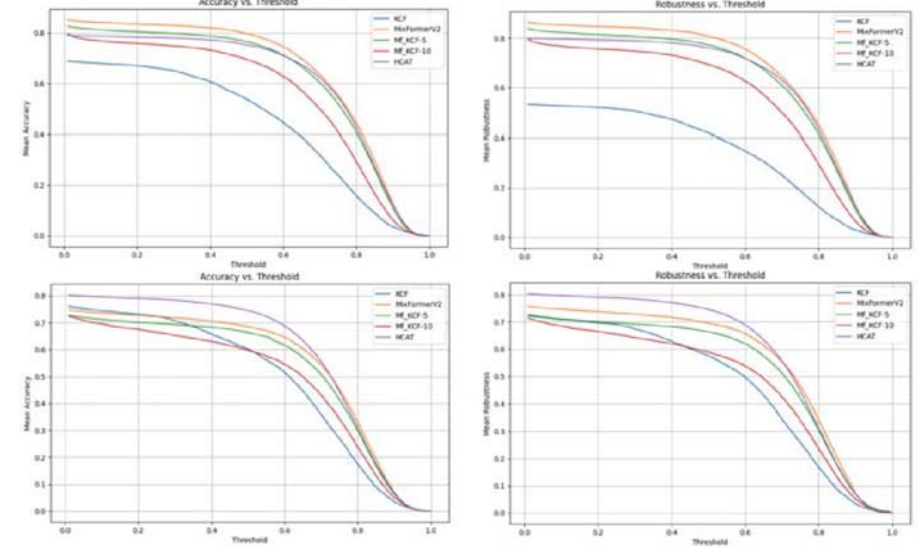


Fig. 3. ROC-curves for accuracy and robustness on UAV123 (top) and VisDrone-SOT (bottom) datasets (on Jetson Orin)

#### References

- [1] Kristan M. et al. The tenth visual object tracking VOT2022 challenge results // European Conference on Computer Vision. – Cham: Springer Nature Switzerland, 2022. – C. 431-460.
- [2] Bolme, D.S., Beveridge, J.R., Draper, B.A., and Lui, Y.M., 2010. Visual object tracking using adaptive correlation filters. In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (pp. 2544-2550). IEEE.
- [3] Henriques, João F., et al. "High-speed tracking with kernelized correlation filters." IEEE transactions on pattern analysis and machine intelligence 37.3 (2014): 583-596.
- [4] Danelljan, M., Häger, G., Khan, F.S. and Felsberg, M., 2016. Discriminative scale space tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(8), pp. 1561-1575.
- [5] Danelljan, M., Bhat, G., Shahbaz Khan, F. and Felsberg, M., 2016. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V (pp. 472-488). Springer International Publishing.
- [6] Danelljan, M., Bhat, G., Shahbaz Khan, F. and Felsberg, M., 2017. Eco: Efficient convolution operators for tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 6638-6646).
- [7] Lukezic, A., Vojir, T., Zajc, L.C., Matas, J. and Kristan, M., 2018. Discriminative correlation filter tracker with channel and spatial reliability. International Journal of Computer Vision, 126(7), pp. 671-688.
- [8] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. and Uszkoreit, J., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.
- [9] Wang, N., Zhou, W., Wang, J. and Li, H., 2021. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 1571-1580).
- [10] Yan B. et al. Learning spatio-temporal transformer for visual tracking // Proceedings of the IEEE/CVF international conference on computer vision. – 2021. – C. 10448-10457.

- [11]. Chen, X., Yan, B., Zhu, X., Wang, D., Yang, X. and Lu, H., 2022. Efficient visual tracking via hierarchical cross-attention transformer. In European Conference on Computer Vision (pp. 461-477). Cham: Springer Nature Switzerland.
- [12]. Cui, Y., Zhuang, B., Li, Y., Yuan, L., Wu, W. and Lin, L., 2022. Mixformer: End-to-end tracking with iterative mixed attention. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 13608-13618).
- [13]. Cui, Y., Zhuang, B., Li, Y., Yuan, L., Wu, W. and Lin, L., 2024. Mixformerv2: Efficient fully transformer tracking. Advances in Neural Information Processing Systems, 36.
- [14]. Mao H. et al. PatchNet--Short-range Template Matching for Efficient Video Processing // arXiv preprint arXiv:2103.07371. – 2021.
- [15]. Ji Q. et al. Real-time embedded object detection and tracking system in Zynq SoC // EURASIP Journal on Image and Video Processing. – 2021. – Т. 2021. – С. 1-16.
- [16]. Liu W. et al. Ssd: Single shot multibox detector //Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. – Springer International Publishing, 2016. – С. 21-37.
- [17]. Du L., Ho A. T. S., Cong R. Perceptual hashing for image authentication: A survey // Signal Processing: Image Communication. – 2020. – Т. 81. – С. 115713.
- [18]. Mueller, Matthias et al. “A Benchmark and Simulator for UAV Tracking.” European Conference on Computer Vision (2016).
- [19]. Zhu, P., Wen, L., Bian, X., Haibin, L. and Hu, Q., 2021. Detection and tracking meet drones challenge. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(11), pp. 7380-7399.
- [20]. Kristan, M., et al., 2023. The VOTS2023 Challenge Performance Measures.
- [21]. Official implementation of the Hierarchical Cross-Attention Transformer (HCAT) tracker: <https://github.com/chenxin-dlut/HCAT>
- [22]. MixFormerV2 implementation with CUDA, TensorRT and Onnx: <https://github.com/maliangzhibi/Mixformerv2-onnx>
- [23]. ViT tracker from OpenCV\_zoo: [https://github.com/opencv/opencv\\_zoo/tree/main/models/object\\_tracking\\_vittrack](https://github.com/opencv/opencv_zoo/tree/main/models/object_tracking_vittrack)

### ***Информация об авторах / Information about authors***

Армен САРДАРЯН получил степень бакалавра в области прикладной математики и информатики в Санкт-Петербургском государственном университете, Россия, в 2021 году. В 2024 он получил степень магистра в области интеллектуальных систем и робототехники в Российско-Армянском университете, Армения. В настоящее время является аспирантом по направлению “Математическое моделирование, численные методы и комплексы программ” в Российско-Армянском университете, Армения. Он также является исследователем в Центре Передовых Программных Технологий (CAST). Его исследовательские интересы включают БПЛА, глубокое обучение, компьютерное зрение и анализ данных.

Armen SARDARYAN received his B.Sci. degree in Applied Mathematics and Computer Science from St. Petersburg State University, Russia, in 2021. He obtained a M.Sc. degree in Intellectual Systems and Robotics from Russian-Armenian University, Armenia, in 2024. He is currently a PhD student in Mathematical Modeling, Numerical Methods and Program Complexes at the Russian-Armenian University, Armenia. He is also a researcher at the Center for Advanced Software Technologies (CAST). His research interests include UAVs, deep learning, computer vision and data analysis.

Вардан СААКЯН – научный сотрудник Центра передовых программных технологий (CAST) и аспирант Российско-Армянского университета, специализируется на математическом и программном обеспечении вычислительных систем. Получил степень бакалавра по информатике и прикладной математике в Национальном политехническом университете Армении (2021) и степень магистра по интеллектуальным системам и робототехнике в Российско-Армянском университете (2023). Его исследования посвящены беспилотным летательным аппаратам, компьютерному зрению и обучению с подкреплением.

Vardan SAHAKYAN is a researcher at the Center of Advanced Software Technologies (CAST) and a postgraduate student at Russian-Armenian University, specializing in mathematical and software support for computing systems. He holds a B.Sci. in informatics and applied mathematics from the National Polytechnic University of Armenia (2021) and an M.Sc. in intellectual systems and robotics from Russian-Armenian University (2023). His research focuses on UAVs, computer vision, and reinforcement learning.

Ваагн МЕЛКОНЯН получил степень бакалавра в области информатики и прикладной математики в Национальном Политехническом Университете Армении, Армения, в 2021 году. В 2023 году он получил степень магистра в области интеллектуальных систем и робототехники в Российско-Армянском Университете, Армения. В настоящее время он занимается аспирантурой по математическому и программному обеспечению вычислительных машин, комплексов и компьютерных сетей в Российско-Армянском Университете, Армения. Он также является исследователем в Центре Передовых Программных Технологий (CAST). Его исследовательские интересы включают БПЛА, компьютерное зрение и алгоритмы управления.

Vahagn MELKONYAN received his B.Sc. in Informatics and Applied Mathematics from the National Polytechnic University of Armenia, Armenia, in 2021. In 2023, he earned his M.Sc. degree in Intellectual Systems and Robotics from Russian-Armenian University, Armenia. He is currently pursuing a Ph.D. in Mathematical and Software Support for Computing Machines, Complexes, and Computer Networks at Russian-Armenian University, Armenia. He is also a researcher at the Center of Advanced Software Technologies (CAST). His research interests include UAVs, computer vision, and control algorithms.

Севак САРГСЯН получил степени бакалавра и магистра в области информатики и прикладной математики в Ереванском Государственном Университете, Армения, в 2010 и 2012 годах соответственно. Позже, в 2016 году, он получил степень кандидата физ.-мат. наук в области математического и программного обеспечения вычислительных машин, комплексов и компьютерных сетей в Институте системного программирования имени Иваницова Российской академии наук. В настоящее время он является заведующим кафедрой Системного Программирования в Российско-Армянском Университете, Армения. Его исследовательские интересы включают технологии компиляторов, безопасность программного обеспечения и тестирование программного обеспечения.

Sevak SARGSYAN received his B. Sci. and M. Sci. degrees in informatics and applied mathematics from Yerevan State University, Armenia, in 2010 and 2012, respectively. He later in 2016 obtained his Cand. Sci. (Phys.-Math.) degree in mathematical and software support for computing machines, complexes, and computer networks from the Ivannikov Institute for System Programming of the Russian Academy of Sciences. Presently he serves as the head of the system programming department at Russian-Armenian University, Armenia. His research interests include compiler technologies, software security, and software testing.