ORES

Научно-практический электронный журнал
**Оригинальные исследования (ОРИС)**, *ISSN 2222-0402*
www.ores.su
original-research.ru

**УДК 004.932.2**

# ОБЗОР И СРАВНЕНИЕ АЛГОРИТМОВ ТРЕКИНГА С ПОМОЩЬЮ БИБЛИОТЕКИ OPENCV

**Вершинин Евгений Владимирович**
Кандидат физико-математических наук, доцент и заведующий кафедрой «Системы обработки информации»
Калужский филиал Московского государственного технического университета имени Н.Э. Баумана
vershinin@bmstu.ru

**Фадеев Вячеслав Олегович**
Студент группы ИУК5-61Б
Калужский филиал Московского государственного технического университета имени Н.Э. Баумана
fadeevvo@student.bmstu.ru

**Бураков Илья Игоревич**
Студент магистратуры ИУК5-41М
Калужский филиал Московского государственного технического университета имени Н.Э. Баумана
burakovii@student.bmstu.ru

**Аннотация**

В статье рассмотрены различные алгоритмы трекинга объектов, доступных в OpenCV, с целью выявления их преимуществ и недостатков в различных сценариях использования. Также представлена оценка их эффективности, точности и производительность на различных наборах данных в виде сравнительного анализа.

**Ключевые слова:** OpenCV, алгоритм трекинга объектов, компьютерное зрение, искусственный интеллект.

# OVERVIEW AND COMPARISON OF TRACKING ALGORITHMS USING OPENCV LIBRARY

**Evgeny V. Vershinin**
Ph.D, Associate Professor and Head of the department "Information Processing Systems"
Bauman Moscow State Technical University (Kaluga Branch
vershinin@bmstu.ru

**Vyacheslav O. Fadeev**
Student of group IUK5-61B
Bauman Moscow State Technical University (Kaluga Branch
fadeevvo@student.bmstu.ru

**Ilya I. Burakov**
Student of group IUK5-41M
Bauman Moscow State Technical University (Kaluga Branch
burakovii@student.bmstu.ru

ABSTRACT

This article examines the various object tracking algorithms available in OpenCV in order to identify their advantages and disadvantages in various use cases. An assessment of their effectiveness, accuracy and performance on various data sets is also presented in the form of a comparative analysis.

**Keywords:** OpenCV, object tracking algorithm, computer vision, artificial intelligence.

Введение

В современном мире компьютерное зрение становится все более важным в различных областях, начиная от автоматизации производства до развлекательной индустрии. Одним из ключевых аспектов компьютерного зрения является трекинг объектов - процесс непрерывного отслеживания движущихся объектов на видео. С его помощью можно решать широкий спектр задач, начиная от слежения за объектами на дорогах до отслеживания движений в видеоиграх.

В данной статье мы сосредоточимся на сравнении алгоритмов трекинга объектов, реализованных с использованием библиотеки OpenCV на языке программирования Python. OpenCV (Open Source Computer Vision Library) является одним из наиболее популярных инструментов в области компьютерного зрения благодаря своей мощности, гибкости и простоте использования.

Целью является сравнение различных алгоритмов трекинга объектов, доступных в OpenCV, с целью выявления их преимуществ и недостатков в различных сценариях. Мы оценим их эффективность, точность и производительность на различных наборах данных, а также проведем сравнительный анализ результатов.

Полученные в результате данные могут быть полезны для разработчиков, исследователей и специалистов в области компьютерного зрения для выбора наиболее подходящего алгоритма трекинга объектов в конкретных приложениях.

Библиотека OpenCV

OpenCV (Open Source Computer Vision Library) — это библиотека с открытым исходным кодом, разработанная для обработки изображений и компьютерного зрения в реальном времени. Она предоставляет широкий спектр функций и алгоритмов, которые позволяют выполнять задачи, связанные с обработкой изображений, распознаванием образов, анализом видео и многими другими приложениями в области компьютерного зрениях [1].

Основные возможности OpenCV включают в себя [2]:

1. Загрузка и сохранение изображений и видео: OpenCV поддерживает множество форматов файлов изображений и видео, что обеспечивает гибкость при работе с различными источниками данных.

2. Обработка изображений: Библиотека предоставляет множество функций для изменения размера, поворота, наложения фильтров, обрезки и других операций с изображениями.

3. Обнаружение и распознавание объектов: OpenCV включает в себя алгоритмы для обнаружения объектов на изображениях, такие как лица, люди, автомобили и другие объекты, а также для распознавания образов на изображениях.

4. Вычисление оптического потока: Этот функционал позволяет анализировать движение объектов на видео и извлекать информацию о скорости и направлении движения.

5. Машинное обучение и глубокое обучение: OpenCV интегрируется с популярными библиотеками машинного обучения и глубокого обучения, такими как TensorFlow и PyTorch, для создания и обучения моделей компьютерного зрения.

6. Работа с камерами: OpenCV предоставляет возможности для захвата и обработки видеопотока с веб-камер и других устройств захвата изображений.

Благодаря своей мощности и гибкости OpenCV является одной из наиболее популярных библиотек в области компьютерного зрения и находит применение в различных областях, включая медицину, робототехнику, автомобильную промышленность, безопасность и многое другое [2].

В работе сравниваются две версии: 4.5.5 и 4.7.0.

Алгоритмы

BOOSTING Tracker — это один из алгоритмов трекинга объектов, реализованных в библиотеке OpenCV на языке программирования Python. Этот алгоритм основан на адаптивном ансамбле усиления (Adaboost), который комбинирует несколько слабых классификаторов для создания сильного классификатора. BOOSTING Tracker предоставляет простой и эффективный метод для отслеживания объектов на видео [3].

Принцип работы BOOSTING Tracker заключается в создании ансамбля слабых классификаторов, которые последовательно адаптируются к изменяющимся характеристикам объекта. Этот алгоритм основан на идее, что каждый слабый классификатор вносит свой вклад в общий результат, а комбинация всех слабых классификаторов обеспечивает высокую точность трекинга.

Одним из основных преимуществ BOOSTING Tracker является его способность к адаптации к изменяющимся условиям освещения, масштаба и поворота объекта. Это делает его подходящим для трекинга объектов в различных сценариях, включая условия с непостоянным освещением или изменяющимся фоном.

Однако, несмотря на свою эффективность, BOOSTING Tracker может столкнуться с некоторыми ограничениями, такими как чувствительность к шуму или низкая устойчивость к быстро изменяющимся характеристикам объекта. Поэтому в некоторых сценариях использование других алгоритмов трекинга может быть более предпочтительным.

MIL Tracker (Multiple Instance Learning Tracker) - еще один алгоритм трекинга объектов, доступный в библиотеке OpenCV на Python. Этот алгоритм основан на принципе обучения с множественными примерами (Multiple Instance Learning), который позволяет эффективно трекировать объекты на видео, учитывая их изменчивость и разнообразие в различных кадрах [3].

Принцип работы MIL Tracker состоит в том, чтобы рассматривать объект как совокупность множества «экземпляров» (instances), каждый из которых может быть как

положительным, так и отрицательным примером. Алгоритм обучается на этой совокупности экземпляров, что позволяет ему учитывать различные варианты внешнего вида объекта.

Одним из ключевых преимуществ MIL Tracker является его способность обучаться на ходу. Это означает, что алгоритм может адаптироваться к изменяющимся условиям трекинга и динамически обновлять свою модель объекта, что повышает его точность и устойчивость к различным сценариям.

Однако, MIL Tracker также имеет свои ограничения. Например, он может быть менее эффективным в условиях сильного изменения масштаба или поворотов объекта, поскольку не всегда способен адекватно адаптироваться к таким изменениям.

KCF Tracker (Kernelized Correlation Filters Tracker) — это еще один из алгоритмов трекинга объектов, реализованных в библиотеке OpenCV на Python. Этот алгоритм основан на применении ядерных фильтров корреляции для эффективного и точного трекинга объектов на видео [4].

Принцип работы KCF Tracker заключается в использовании ядерных методов для вычисления корреляции между объектом, который необходимо отследить, и окрестностями его предполагаемой позиции на последующих кадрах видео. Это позволяет алгоритму добиться высокой точности и устойчивости к различным условиям освещения, масштабирования и поворотов объекта.

Одним из ключевых преимуществ KCF Tracker является его высокая скорость работы. Благодаря оптимизированной реализации и использованию ядерных методов для вычисления корреляции, этот алгоритм способен работать в реальном времени, что делает его подходящим для широкого спектра приложений, включая системы видеонаблюдения, автономные транспортные средства и многое другое.

Однако, как и у любого алгоритма, у KCF Tracker есть свои ограничения. Например, он может быть менее эффективным в условиях сильных изменений объекта, таких как деформации или перекрытия с другими объектами.

TLD Tracker (Tracking-Learning-Detection) — это алгоритм трекинга объектов, разработанный для комбинирования процессов трекинга, обучения и детектирования для повышения точности и устойчивости трекинга в различных условиях [4].

Основная идея TLD Tracker состоит в том, чтобы использовать адаптивное обучение для коррекции и улучшения результатов трекинга на каждом новом кадре видео. Алгоритм состоит из трех основных этапов: трекинга (tracking), обучения (learning) и детектирования (detection).

На этапе трекинга TLD Tracker использует методы отслеживания объекта по кадру, применяя фильтры и классификаторы для определения положения объекта на следующем кадре. Затем, на этапе обучения, алгоритм анализирует результаты трекинга и использует их для обучения модели объекта. Наконец, на этапе детектирования TLD Tracker использует обученную модель для определения объекта на следующем кадре, что позволяет ему обнаруживать объекты, даже если они были временно потеряны в процессе трекинга.

Одним из основных преимуществ TLD Tracker является его способность к обучению на ходу, что позволяет ему адаптироваться к изменяющимся условиям сцены и повышает его устойчивость к различным видам возмущений. Кроме того, он обеспечивает высокую точность детектирования и трекинга объектов даже в условиях сильных изменений освещения, масштабирования и фона.

Однако, TLD Tracker также имеет свои ограничения, включая потребность в дополнительных вычислительных ресурсах для обучения и детектирования объектов, что может сделать его менее подходящим для ресурсоемких приложений в реальном времени.

Алгоритм трекинга MEDIANFLOW (Медианный поток) является популярным и эффективным методом отслеживания объектов в видео. Он основан на анализе движения объектов в последовательности кадров и определении наиболее вероятной траектории движения на основе медианы смещений пикселей между кадрами [4].

Принцип работы MEDIANFLOW Tracker заключается в вычислении медианного вектора смещения между текущим и предыдущим кадром для каждого объекта. Затем используется эта информация для коррекции позиции объекта на следующем кадре. Поскольку медианный вектор смещения устойчив к выбросам и шумам, а также хорошо отражает общее направление движения объекта, MEDIANFLOW Tracker демонстрирует хорошую производительность в различных сценариях, включая ситуации с быстрым движением объектов или изменением их масштаба.

Одним из ключевых преимуществ MEDIANFLOW Tracker является его скорость работы и устойчивость к изменениям внешнего окружения. Благодаря простоте алгоритма и низким вычислительным требованиям, этот метод отслеживания объектов может быть использован в реальном времени на устройствах с ограниченными ресурсами.

Однако, как и у любого алгоритма трекинга, у MEDIANFLOW Tracker есть свои ограничения. Например, он может быть менее эффективным в условиях сильного изменения освещения или фона, а также при перемещении объекта за пределы видимости камеры на несколько кадров.

GOTURN (Generic Object Tracking Using Regression Networks) — это алгоритм трекинга объектов, который использует глубокие нейронные сети для предсказания смещения объекта между кадрами видео. В библиотеке OpenCV для Python также доступен этот алгоритм, позволяя разработчикам легко внедрять его в свои проекты [5].

Основной идеей GOTURN является использование нейронных сетей для обучения модели, которая может предсказывать смещение объекта на следующем кадре. Для обучения используются пары изображений, содержащих исходное изображение и его корректную аннотацию (прямоугольник, ограничивающий объект). После обучения нейронная сеть может использоваться для отслеживания объектов на новых кадрах, предсказывая их положение на следующем кадре.

Одним из главных преимуществ GOTURN является его точность трекинга, особенно при быстром движении объектов или изменениях в их внешнем виде. Это достигается за счет использования глубоких нейронных сетей, которые обучаются на больших объемах данных и способны выявлять сложные закономерности в движении объектов.

Тем не менее, GOTURN также имеет свои ограничения. Например, он может потреблять больше вычислительных ресурсов по сравнению с более простыми алгоритмами трекинга, что может сделать его менее подходящим для использования в реальном времени на устройствах с ограниченными ресурсами.

Алгоритм трекинга MOSSE (Minimum Output Sum of Squared Error) — это легковесный и быстрый метод отслеживания объектов, который использует фильтры корреляции для определения позиции объекта на последующих кадрах видео. Этот алгоритм был разработан для обеспечения высокой скорости работы и хорошей устойчивости к различным условиям сцены [5].

Принцип работы MOSSE Tracker основан на использовании адаптивных фильтров корреляции для поиска объекта на следующем кадре. Эти фильтры корреляции обучаются на начальном кадре видео, чтобы создать модель объекта. Затем, на следующих кадрах, алгоритм использует эту модель для определения позиции объекта, вычисляя смещение объекта относительно его предполагаемого положения.

Одним из основных преимуществ MOSSE Tracker является его высокая скорость работы и низкие вычислительные требования. Благодаря простоте алгоритма и

эффективному использованию фильтров корреляции, MOSSE Tracker может быть использован в реальном времени на устройствах с ограниченными ресурсами.

Однако, MOSSE Tracker также имеет свои ограничения. Например, он может быть менее эффективным в условиях сильного изменения освещения, масштабирования объекта или изменения его внешнего вида.

Алгоритм трекинга CSRT (Channel and Spatial Reliability Tracker) — это высокоточный и устойчивый метод отслеживания объектов, который был разработан для обеспечения точного и надежного трекинга объектов в видео.

Принцип работы CSRT Tracker заключается в использовании комбинации двух наборов признаков: канальных (цветовых) и пространственных (пространственных признаков), а также внедрении механизмов, обеспечивающих надежность отслеживания [5].

Канальные признаки предоставляют информацию о цвете и текстуре объекта, а пространственные признаки описывают его форму и контур. CSRT Tracker комбинирует эти признаки для построения модели объекта и использует ее для отслеживания объекта на последующих кадрах видео.

Одним из основных преимуществ CSRT Tracker является его высокая точность трекинга. Благодаря использованию двух наборов признаков и механизмов, обеспечивающих надежность отслеживания, этот алгоритм способен успешно справляться с различными условиями сцены, включая изменения освещения, масштаба и внешнего вида объекта.

Кроме того, CSRT Tracker также обладает высокой скоростью работы и низкими вычислительными требованиями, что позволяет его использовать в реальном времени на устройствах с ограниченными ресурсами.

Аппаратная конфигурация для проведения сравнительного анализа

Сравнительный анализ был выполнен на следующей аппаратной конфигурации:

- Процессор: AMD Ryzen 7 2700 @3.2 ГГц
- Оперативная память: 48 Гб
- Операционная система: Ubuntu 22.04 LTS
- Видеокарта: RTX 3060 TI 8 Гб

Фотография бокса за котором будет следить алгоритм представлен на рисунке 1.

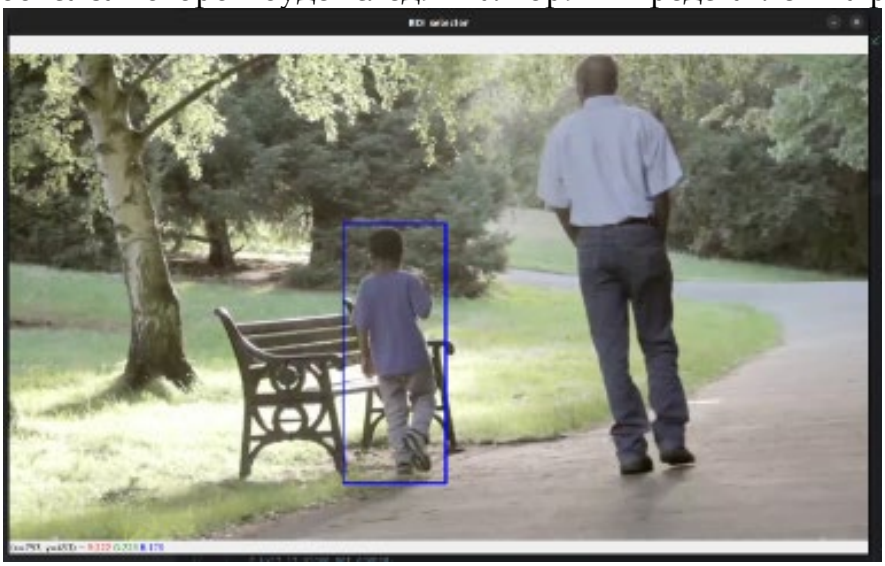

Рисунок 1 - Фотография бокса за котором будет следить алгоритм (Источник: https://www.redcoolmedia.net/download/videos/science/video-off-the-grid-tech-presents-clearunited-clearphone)

Результаты сравнительного анализа с использованием библиотеки OpenCV версии 4.5.5

Для сравнения алгоритмов были использованы следующие критерии:

FPS - кадровая частота или количество сменяемых кадров за единицу времени.

ms- среднее время в миллисекундах (ms), которое требуется на обработку одного кадра изображения для каждого алгоритма.

В Таблице 1 представлены результаты сравнительного анализа с использованием библиотеки OpenCV версии 4.5.5.

Таблица 1. Производительность алгоритмов версии библиотеки 4.5.5

| Алгоритм | Версия CV | CPU | Пямять, МБ | Среднее значение |
|---|---|---|---|---|
| BOOSTING | 4.5.5 | 3-4 ядра | RES: 300-400 | FPS: 39.79 ms: 25.9 |
| MIL | 4.5.5 | 7-8 ядра | RES:200-300 | FPS: 20.1 ms:50.47 |
| KCF | 4.5.5 | 2-3 ядра | RES:300-350 | FPS: 31.72 ms: 32.05 |
| TLD | 4.5.5 | 6-7 ядра | RES: 200-250 | FPS: 24.02 ms: 41.93 |
| MEDIANFLOW | 4.5.5 | 6-7 ядра | RES: 180-185 | FPS: 272.46 ms: 4.18 |
| GOTURN | 4.5.5 | 8-10 ядра | RES: 600-650 | FPS: 27.16 ms: 41.8 |
| MOSSE | 4.5.5 | 4 ядра | RES: 200 | FPS: 332.73 ms:3.31 |
| CSRT | 4.5.5 | 5-6 ядра | RES: 250-300 | FPS: 29.23 ms:36.82 |

Для удобства значения, также представленные в виде графиков: алгоритм BOOSTING (рисунок 2), алгоритм MIL (рисунок 3), алгоритм KCF (рисунок 4), алгоритм TLD (рисунок 5), алгоритм MEDIANFLOW (рисунок 6), алгоритм GOTURN (рисунок 7), алгоритм MOSSE (рисунок 8), алгоритм CSRT (рисунок 9).

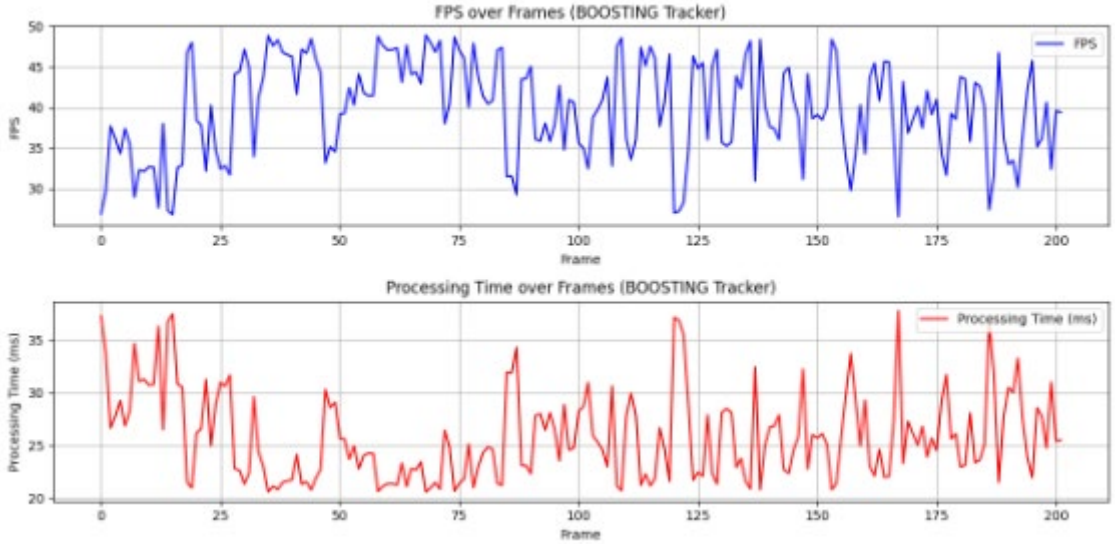


Рисунок 2 – График сравнения алгоритма BOOSTING

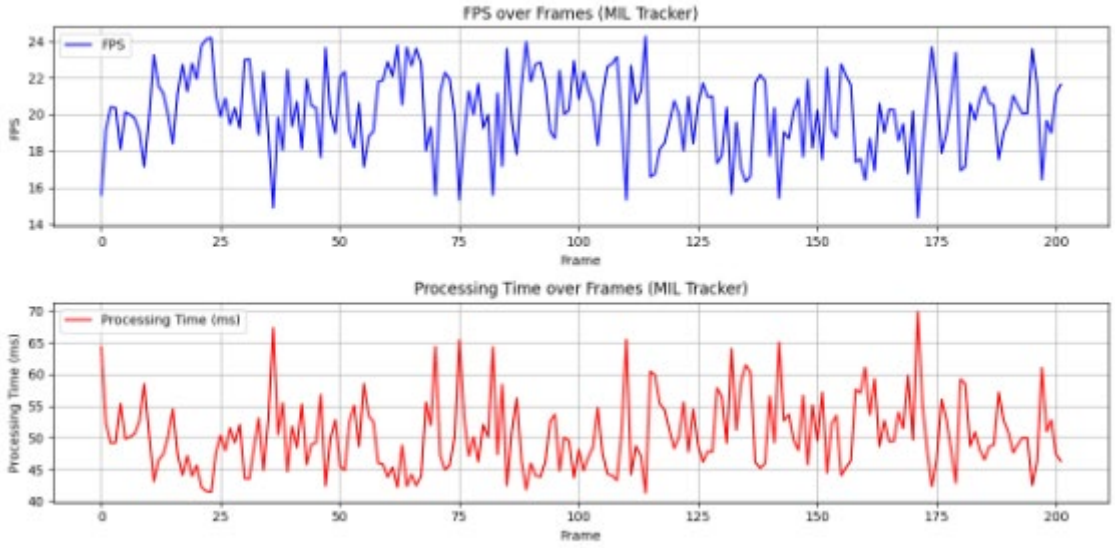


Рисунок 3 – График сравнения алгоритма MIL

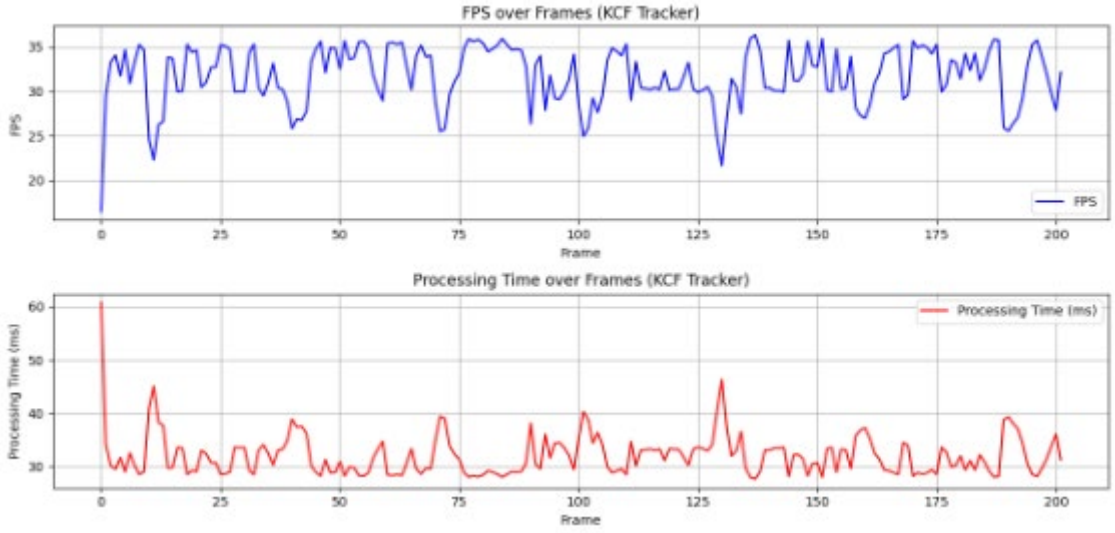


Рисунок 4 – График сравнения алгоритма KCF

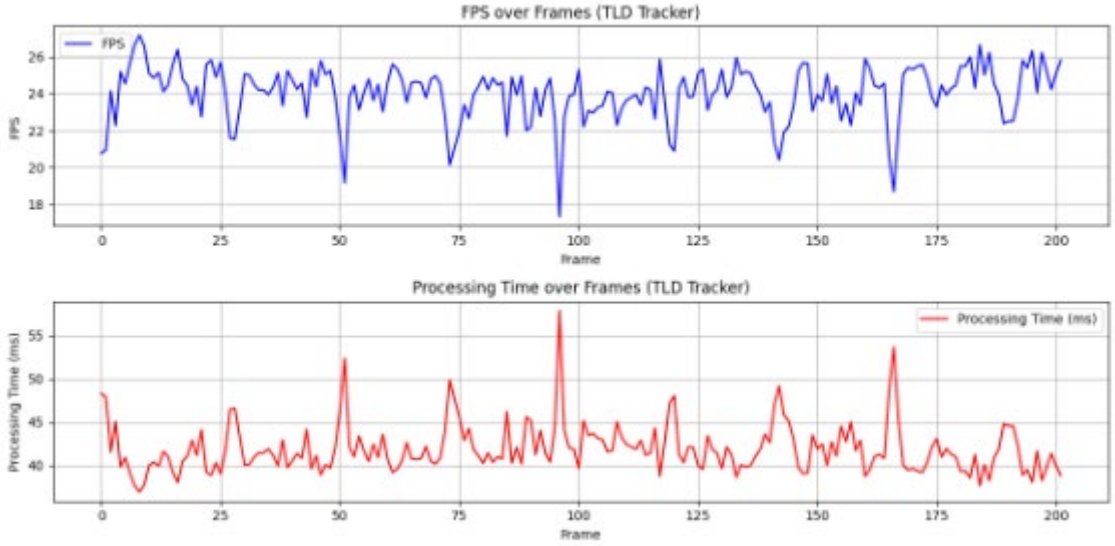


Рисунок 5 – График сравнения алгоритма TLD

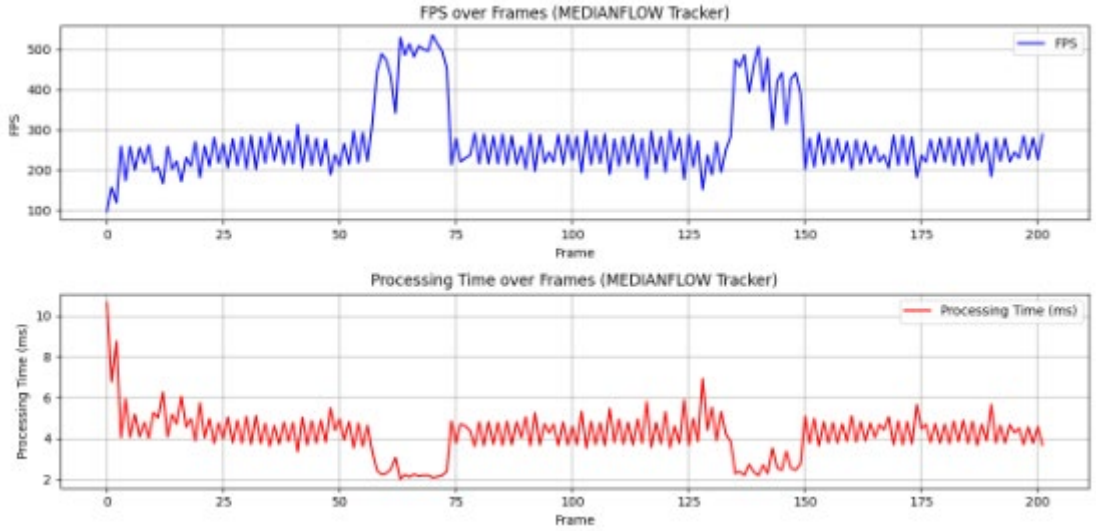


Рисунок 6 – График сравнения алгоритма MEDIANFLOW

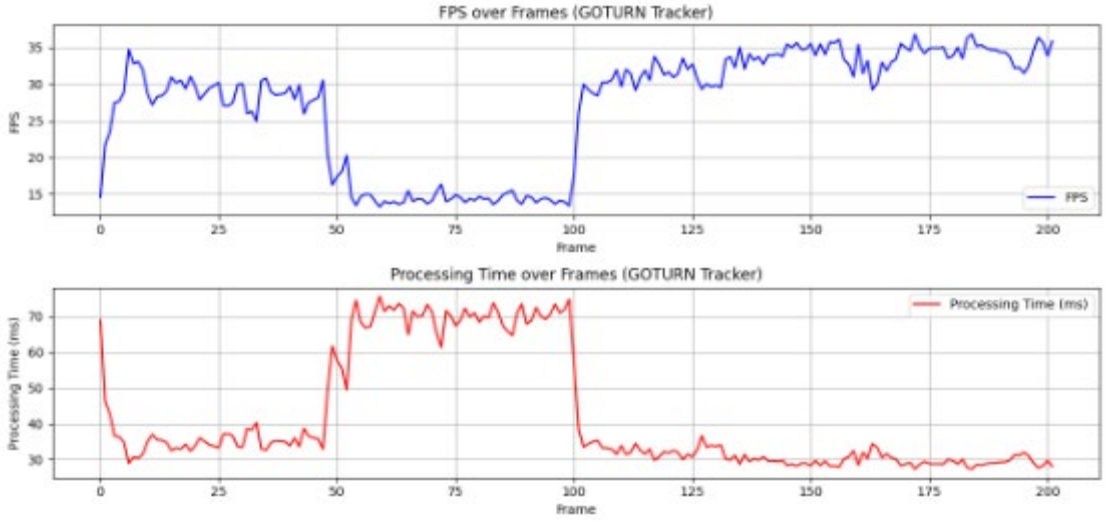


Рисунок 7 – График сравнения алгоритма GOTURN

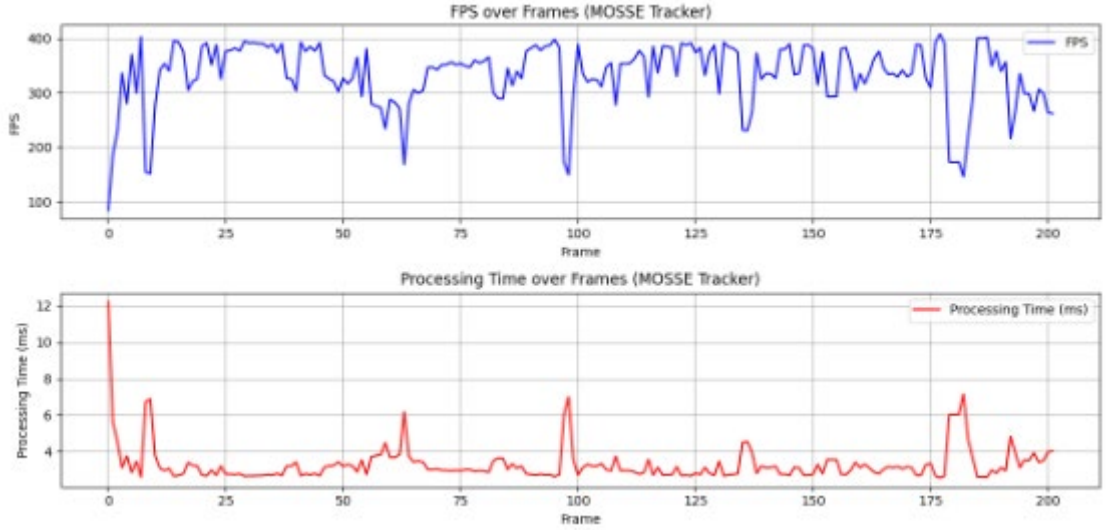


Рисунок 8 – График сравнения алгоритма MOSSE

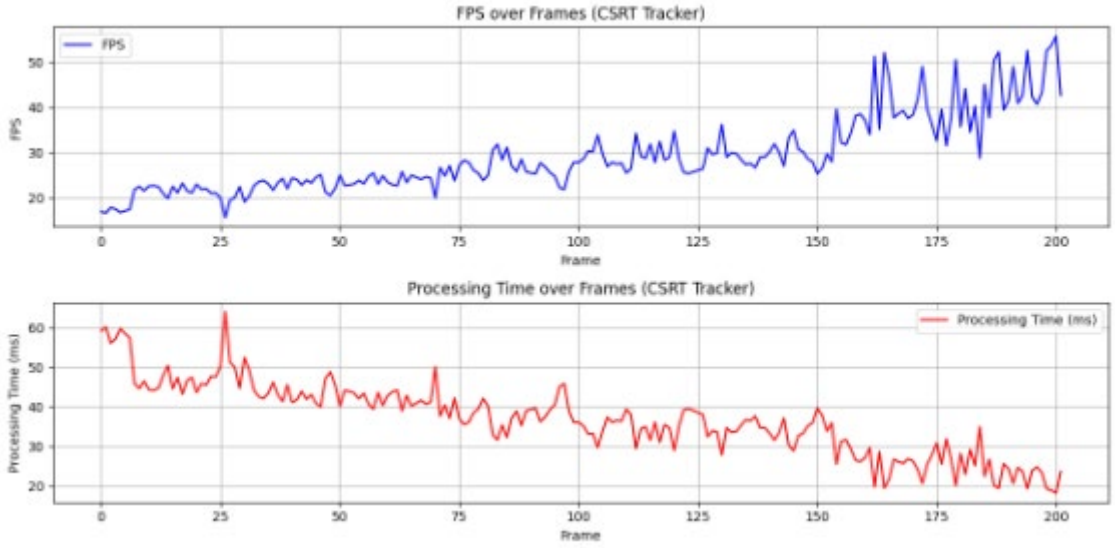


Рисунок 9 – График сравнения алгоритма CSRT

Результаты сравнительного анализа с использованием библиотеки OpenCV версии 4.7.0

В Таблице 2 представлены результаты сравнительного анализа с использованием библиотеки OpenCV версии 4.7.0

Таблица 2. Производительность алгоритмов версии библиотеки 4.7.0

| Алгоритм | Версия CV | CPU | MEM | Среднее значение |
|---|---|---|---|---|
| BOOSTING | 4.7.0 | 3-4 ядра | RES: 300-400 | FPS: 31.65 ms: 32.75 |
| MIL | 4.7.0 | 7-8 ядра | RES:200-300 | FPS: 19.84 ms:51.15 |
| KCF | 4.7.0 | 2-3 ядра | RES:300-350 | FPS: 32.38 ms: 31.41 |
| TLD | 4.7.0 | 6-7 ядра | RES: 200-250 | FPS: 22.14 ms: 45.67 |
| MEDIANFLOW | 4.7.0 | 4-5 ядра | RES: 200 | FPS: 264.92 ms: 4.34 |
| GOTURN | 4.7.0 | 8-10 ядра | RES: 1100 | FPS: 26.33 ms: 43.75 |
| MOSSE | 4.7.0 | 2-3 ядра | RES: 200 | FPS: 344.33 ms:3.12 |
| CSRT | 4.7.0 | 4-6 ядра | RES: 200-250 | FPS: 23.03 ms:46.06 |

Для удобства значения, также представленные в виде графиков: алгоритм BOOSTING (рисунок 10), алгоритм MIL (рисунок 11), алгоритм KCF (рисунок 12), алгоритм TLD (рисунок 13), алгоритм MEDIANFLOW (рисунок 14), алгоритм GOTURN (рисунок 15), алгоритм MOSSE (рисунок 16), алгоритм CSRT (рисунок 17).

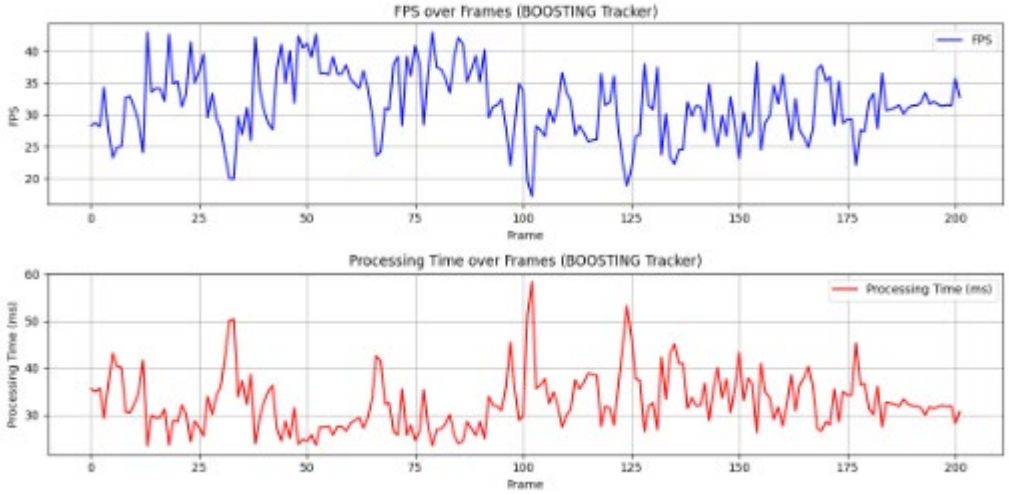


Рисунок 10 – График сравнения алгоритма BOOSTING

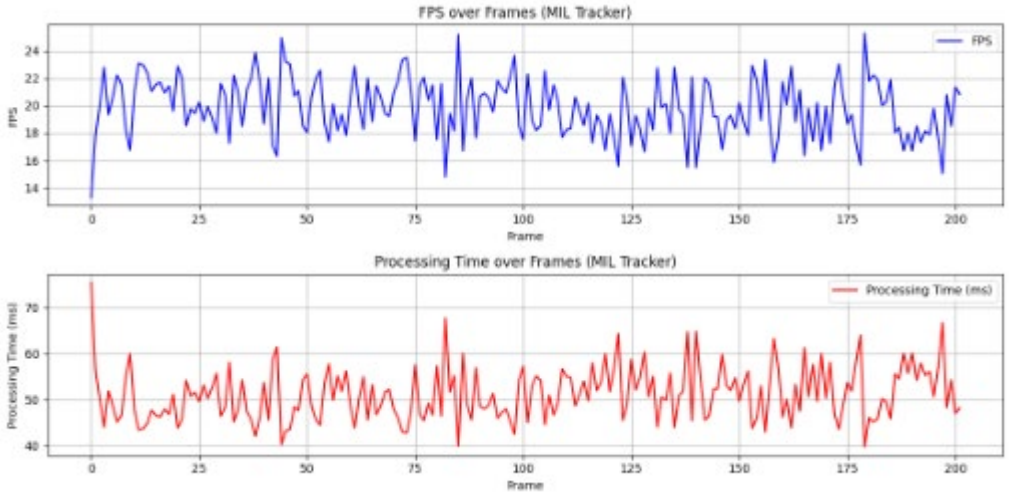


Рисунок 11 – График сравнения алгоритма MIL

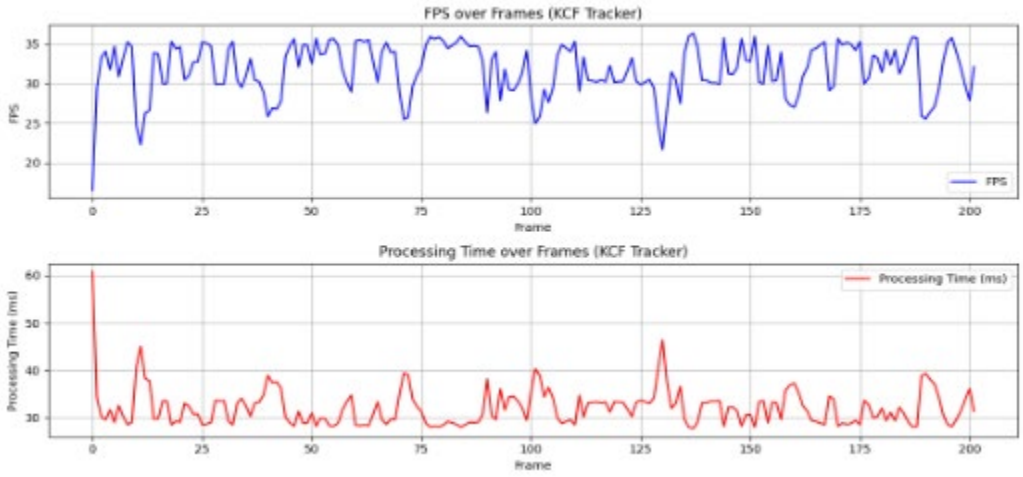


Рисунок 12 – График сравнения алгоритма KCF

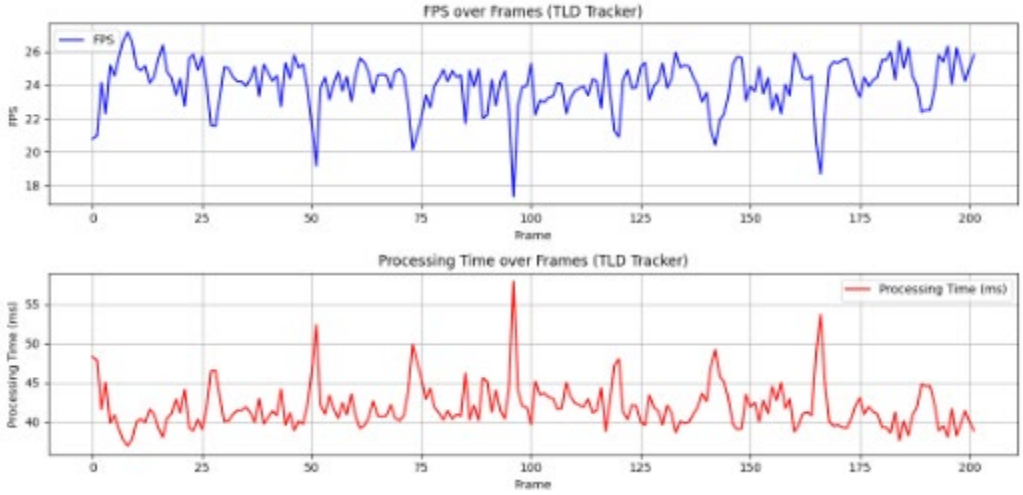


Рисунок 13 – График сравнения алгоритма TLD

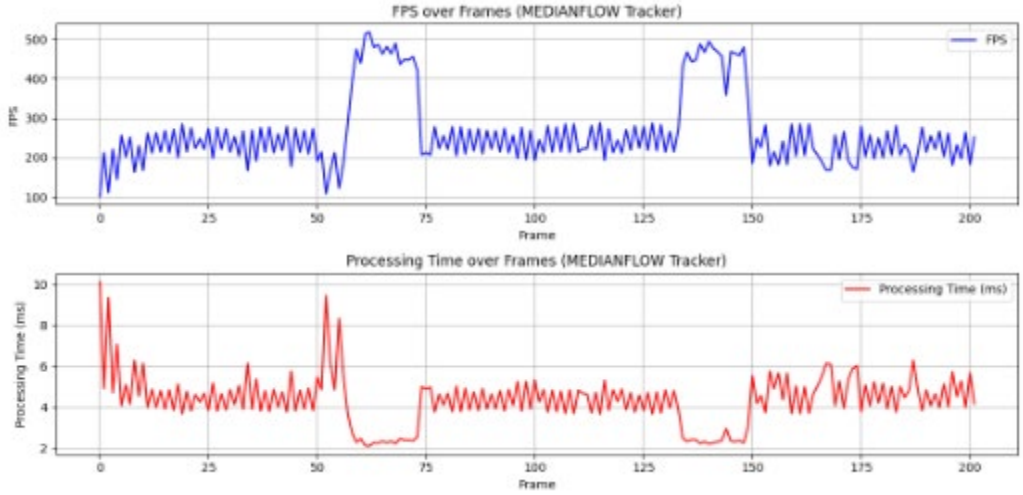


Рисунок 14 – График сравнения алгоритма MEDIANFLOW

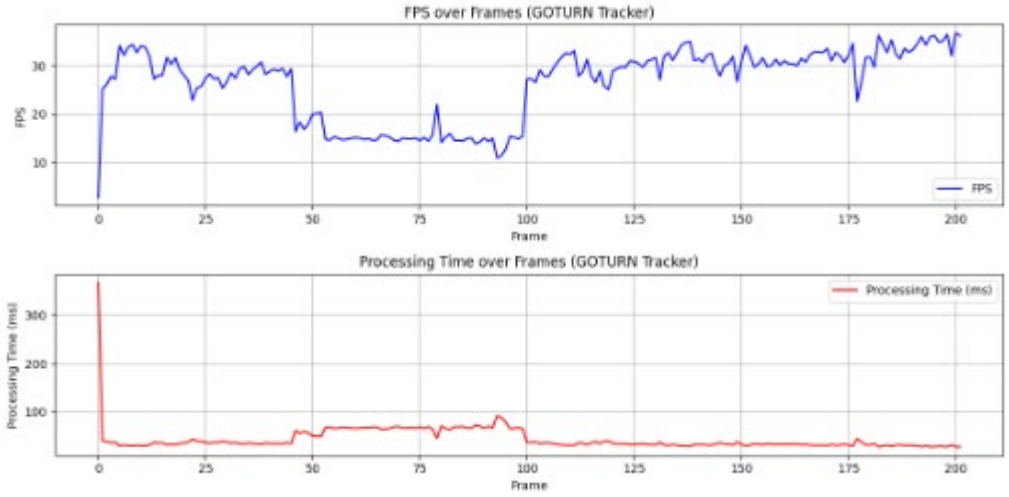


Рисунок 15 – График сравнения алгоритма GOTURN

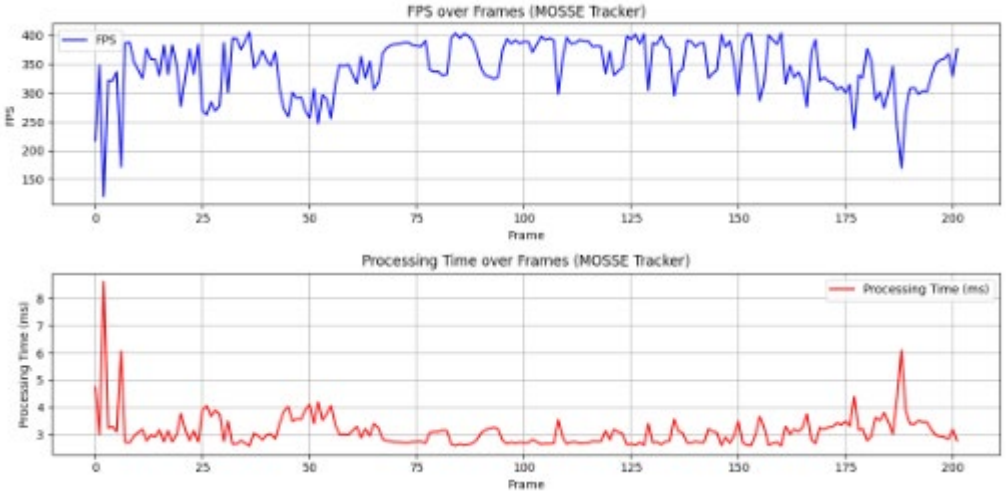


Рисунок 16 – График сравнения алгоритма MOSSE

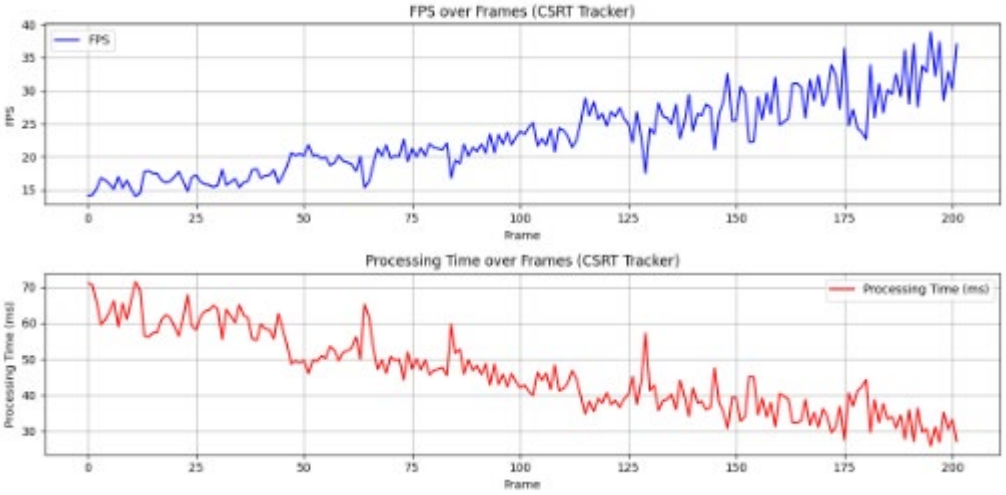


Рисунок 17 – График сравнения алгоритма CSRT

Заключение

В результате проведенного сравнительного анализа алгоритмов трекинга, реализованных с помощью библиотеки OpenCV версий: 4.5.5 и 4.7.0, можно сделать следующие выводы:

Алгоритмы MEDIANFLOW и MOSSE, показывают высокие значения FPS и время, которое требуется на обработку одного кадра, на обеих версиях библиотеки, что указывает на их эффективность.

Алгоритмы MIL и TLD, значения FPS и время, которое требуется на обработку одного кадра, немного хуже на версии 4.7.0 по сравнению с 4.5.5.

Алгоритмы BOOSTING и CSRT, значения FPS и время, которое требуется на обработку одного кадра, примерно одинаковы на обеих версиях.

**Список литературы:**

1. Satya Mallick - Object Tracking using OpenCV (C++/Python) [Электронный ресурс] URL: https://learnopencv.com/object-tracking-using-opencv-cpp-python/ (дата обращения 01.03.2024)
2. A. Sarkar, S. Negi and A. Dangi, "Comparison of Different Tracking Algorithms in OpenCV", International Journal for Research in Applied Science and Engineering Technology, vol. 10, no. 12, pp. 596-598, 2022. [Электронный ресурс] URL: https://www.ijraset.com/best-journal/comparison-of-different-tracking-algorithms-in-opencv (дата обращения 01.03.2024)

3. Object Tracking OpenCV Python [Электронный ресурс] URL: https://hackthedeveloper.com/object-tracking-opencv-python/ (дата обращения 01.03.2024)
4. A Complete Review of the OpenCV Object Tracking Algorithms [Электронный ресурс] URL: https://broutonlab.com/blog/opencv-object-tracking/ (дата обращения 01.03.2024)
5. Adrian Rosebrock - OpenCV Object Tracking [Электронный ресурс] URL: https://pyimagesearch.com/2018/07/30/opencv-object-tracking/ (дата обращения 01.03.2024).

**References:**

1. Satya Mallick - Object Tracking using OpenCV (C++/Python) [Electronic resource] URL: https://learnopencv.com/object-tracking-using-opencv-cpp-python / (accessed 03/01/2024)
2. A. Sarkar, S. Negi and A. Dangi, "Comparison of Different Tracking Algorithms in OpenCV", International Journal for Research in Applied Science and Engineering Technology, vol. 10, No. 12, pp. 596-598, 2022. [Electronic resource] URL: https://www.ijraset.com/best-journal/comparison-of-different-tracking-algorithms-in-opencv (accessed 03/01/2024)
3. Object Tracking OpenCV Python [Electronic resource] URL: https://hackthedeveloper.com/object-tracking-opencv-python / (accessed 03/01/2024)
4. A Complete Review of the OpenCV Object Tracking Algorithms [Electronic resource] URL: https://broutonlab.com/blog/opencv-object-tracking / (accessed 03/01/2024)
5. Adrian Rosebrock - OpenCV Object Tracking [Electronic resource] URL: https://pyimagesearch.com/2018/07/30/opencv-object-tracking / (accessed 03/01/2024)