Разработка алгоритма многообъектного трекинга с необучаемыми признаками сопоставления объектов

В.А. Горбачев [1], В.Ф. Калугин [1]
[1] Государственный научно-исследовательский институт авиационных систем, 125167, Россия, г. Москва, ул. Викторенко, д. 7

Аннотация

Проблема межкадрового сопоставления объектов на видеопоследовательностях (трекинга, отслеживания) множества объектов – одна из самых сложных задач в компьютерном зрении. В данной работе решается задача отслеживания множества объектов на видеозаписях, полученных с борта беспилотного летательного аппарата. Эта задача в отличие от отслеживания со статичной камеры имеет особенность в виде сложного движения и тряски камеры, что приводит к резким изменениям положения точки съёмки, ракурса и масштаба объектов. В этой работе мы исследуем возможность применения улучшения качества трекинга на основе алгоритма ByteTrack, одного из лучших алгоритмов отслеживания для набора данных MOT Challenge, на наборе данных Visdrone 2019.

Ключевые слова: многообъектный трекинг, компенсация движения, дескриптор, фильтр Калмана, Yolo v5, ByteTrack, Visdrone 2019, БЛА.

Цитирование: **Горбачев, В.А.** Разработка алгоритма многообъектного трекинга с необучаемыми признаками сопоставления объектов / В.А. Горбачев, В.Ф. Калугин // Компьютерная оптика. – 2023. – Т. 47, № 6. – С. 1002-1010. – DOI: 10.18287/2412-6179-CO-1275.

Citation: Gorbachev VA, Kalugin VF. Development of a multi-object tracking algorithm with untrained features of object matching. Computer Optics 2023; 47(6): 1002-1010. DOI: 10.18287/2412-6179-CO-1275.

Введение

Задача отслеживания (трекинга) объектов заключается в распознавании объектов и присвоении им уникальных идентификационных номеров, которые сохраняются с объектом на протяжении всего видео. В результате трекинга можно проследить траекторию каждого объекта на всём протяжении его появления на кадрах видеозаписи.

Рассматривают онлайн- и офлайн-трекинг. В случае офлайн-трекинга на вход алгоритма подаётся видеопоследовательность целиком, а при онлайн-трекинге алгоритм обрабатывает каждый кадр последовательно и сразу выдаёт полученный результат.

Задача трекинга решается практически во всех системах, использующих анализ видео. Без использования трекинга невозможно понять, как соотносятся объекты на текущем кадре с объектами на предыдущем кадре, сколько объектов вновь появились, а сколько из них уже были раньше, какова траектория движения объектов. Кроме того, с помощью трекинга возможно исправлять ошибки алгоритма детектирования, то есть фильтровать ложные и находить пропущенные обнаружения.

Мы будем следовать наиболее успешной на сегодняшний день парадигме – отслеживанию по обнаружению (tracking by detection), которая заключается в разделении задачи отслеживания и задачи обнаружения. Подход состоит из нескольких отдельных частей: обнаружение объектов, предсказывание местоположений отслеживаемых объектов из имеющихся траекторий моделью движения, ассоциация (связывание) траекторий отслеживаемых объектов и новых обнаружений. Главное преимущество такого подхода – модульность, возможность улучшать отдельные части алгоритма без изменения остальных, изучать различные комбинации методов.

Целью данной работы было увеличение качества трекинга в части уменьшения количества срывов сопровождения, потери и неправильного сопоставления объектов для создания алгоритма, способного устойчиво выполнять сопровождения подвижных и неподвижных объектов в условиях постоянного движения камеры с возможностью резких рывков, а также возникновения загораживания объектов.

В качестве базового алгоритма взят алгоритм ByteTrack, и проведена его глубокая модернизация на каждом из этапов работы. Модификации позволили поднять точность работы без существенного влияния на общую производительность алгоритма. В предлагаемой реализации в качестве детектора будем использовать YOLO v5, в качестве модели движения – NSA Kalman filter, на этапе ассоциации строится матрица близости и Венгерским алгоритмом производится поиск соответствия новых обнаружений с уже отслеженными объектами.

Для онлайн-отслеживания предложены следующие улучшения в исходный алгоритм:

- компенсация движения камеры;
- модификация фильтра Калмана;
- использование быстрых необучаемых дескрипторов для сравнения внешнего вида;
- штраф за несогласованность направлений движения объектов.

Для офлайн-отслеживания предложены:
- линейная интерполяция, устойчивая к движениям камеры;
- удаление лишних траекторий.

Обзор литературы

Задача анализа движения объектов на видео решается различными способами. Изначально использовались методы, основанные на «вычитании фона» [25], [26] и фильтрации частиц [33]. В этом подходе алгоритм обнаруживал и отслеживал все движущиеся объекты. Но в большинстве задач необходимо отслеживать объекты определенных классов, в том числе и находящихся в покое. Поэтому естественно сначала решать задачу обнаружения объектов, а затем их связывания. Такая парадигма получила название «отслеживание по обнаружению» (tracking-by-detection), которая является наиболее распространённой на сегодняшний день. Подобные методы уверенно лидируют на всех популярных соревновательных наборах данных, в частности, на коллекции Visdrone2019 [34] и наборах данных из соревнований MOTchallenge [30]. Их успех связан в том числе с ростом качества нейросетевых детекторов, таких как YOLO [31], [32], обладающих достаточной точностью и при этом возможностью работать в реальном времени. Подход «отслеживания по обнаружению» на практике проявляет себя значительно лучше и в случае подвижной камеры, так как нет необходимости точно привязывать предыдущие кадры для получения обнаружений.

Одним из первых методов, в котором успешно реализован подход «отслеживание по обнаружению», является SORT [1]. Обнаружение происходило свёрточной нейросетью FasterRCNN [16], координаты объектов с прошлых кадров корректировались моделью движения на основе фильтра Калмана [13], затем по геометрической близости строилась матрица сходства, после чего Венгерским алгоритмом [20] определялись оптимальные пары соответствующих объектов. Такой подход позволил достичь лучших на тот момент результатов по соотношению точности и скорости. Эта идея получила дальнейшее развитие в DeepSORT [3], где было предложено использовать признаки внешнего вида из простой сверточной нейросети в матрице сходства, используя взвешенную сумму расстояния Махалонобиса [27] и косинусное сходство признаков внешнего вида. В FairMOT [4] было предложено извлекать признаки внешнего вида в центре объектов и использовать единую нейросеть для получения обнаружений и вектора признаков внешнего вида путем добавления в нейронную сеть CenterNet [17] дополнительной ветви для генерации вектора признаков внешнего вида. В MaaTrack [5] было предложено использовать компенсацию движения камеры на основе коэффициента корреляции изображений ECC [15] для корректирования координат объектов с прошлых кадров перед предсказаниями фильтром Калмана и использовать алгоритм интерполяции на основе компенсации движения камеры для восстановления пропущенных обнаружений на траектории. В StrongSORT [6] предложена модификация фильтра Калмана NSA из [14] и использование только одного обобщенного вектора внешнего вида (EMA bank), в ByteTrack [2] двухэтапная ассоциация траекторий и обнаружений. В OC-SORT [9] предложили интегрировать в матрицу близости согласованность направлений скоростей сопоставляемых объектов и использовать интерполяцию пропущенных обнаружений для уменьшения погрешности в фильтре Калмана. В TransTrack [7] предложен алгоритм на базе нейросетевой архитектуры Transformer, выполняющей функции модели движения. В SotMOT [8] предложили использовать идею отслеживания одного объекта в задаче отслеживания многих объектов: для каждого объекта обучить модель гребневой регрессии для различения этого объекта и соседних.

Другая парадигма отслеживания заключается в том, чтобы использовать одну нейронную сеть для обнаружения, модели движения и отслеживания. Одним из первых подобных методов, сравнимых по качеству с другими методами отслеживания, был Tracktor [10]. К нейросети FasterRCNN [16] были добавлены два расширения: регрессионная ветвь для имитации модели движения и ветвь для генерации признаков для задачи повторной идентификации. Этот подход получил дальнейшее развитие в Unicorn [11], где предложена архитектура сети, использующая общие признаки внутреннего представления и единую «голову» нейросети для решения четырех задач: однообъектного трекинга (singe object tracking, SOT), сегментации объектов на видеопоследовательностях (video object segmentation, VOS), многообъектного трекинга (multiple object tracking, MOT), многообъектного трекинга с одновременной сегментацией (multiple object tracking segmentation, MOTS).

Метод

Задача трекинга заключается в следующем. На входе в алгоритм трекинга поступает видеопоследовательность $V=\{f_k\}$ и последовательность наборов детекций $\{D_k\}$, соответствующих каждому кадру видеопоследовательности. Детекции выдаются внешним по отношению к алгоритму трекинга алгоритмом *Det* детектирования объектов на изображениях: $D_k=Det(f_k)$. Набор детекций на каждом кадре состоит из кортежей $d_k^l=\left(B_k^l,\ label_k^l,\ score_k^l\right)$, где B – координаты ограничительного прямоугольника объекта, label – класс объекта, score – достоверность обнаружения этого объекта алгоритмом детектирования.

На выходе алгоритм трекинга должен выдавать набор траекторий T. Траектория – это упорядоченная по времени последовательность детекций объектов на различных кадрах, относящихся к одному и тому же

объекту: $T = \{(d_{i_1}^{j_{i1}}, \ldots, d_{i_n}^{j_{in}})\}$, где i_1 – номер кадра начала отслеживания, i_n – номер кадра окончания отслеживания, j_{i1} – номер соответствующей объекту детекции на кадре i_1, j_{in} – номер соответствующей объекту детекции на кадре i_n, $i_1 < i_2 < \ldots < i_n$, $n \geq 2$.

Онлайн-алгоритмы трекинга отличаются от алгоритмов офлайн-отслеживания тем, что онлайн-алгоритмы получают кадры и соответствующие им детекции последовательно, а офлайн – получают всю последовательность сразу.

В качестве базовой модели для решения задачи трекинга мы используем ByteTrack [2] – один из лучших алгоритмов по соотношению скорость-качество для отслеживания на наборах данных из MOT challenge [30]. Алгоритм состоит из трех основных этапов: обнаружение объектов, предсказание новых местоположений объектов на последующих кадрах моделью движения, ассоциация (то есть связывание обнаруженных объектов на новом изображении с уже имеющимися траекториями объектов с предыдущих кадров). Особенностью алгоритма является двухэтапная ассоциация: на первом этапе производится связывание высокодостоверных обнаружений (тех, у которых высокий показатель уверенность детектора в наличии объекта), на втором – связывание обнаружений с низкой достоверностью и несопоставленных на первом этапе траекторий.

В этой статье предлагаются улучшения для всех этапов алгоритма, главными из которых являются улучшения модели движения и ассоциации траекторий и обнаружений, а также добавление четвертого этапа постобработки, содержащего интерполяцию, устойчивую к движениям камеры, для заполнения пропусков в отслеживании и фильтрацию повторяющихся траекторий, которые инициализированы для одного и того же объекта.

Псевдокод алгоритма представлен ниже. На вход алгоритм получает последовательность кадров V, детектор Det, порог τ для достоверности обнаружений.

На первом этапе полученные на каждом кадре от детектора наборы детекций разделяем на высокодостоверные обнаружения D_{high} и низкодостоверные обнаружения D_{low}. Для высокодостоверных $score > \tau$.

На втором этапе применяем модель движения, состоящую из фильтра Калмана и функции компенсации движения камеры, которые формируют прогноз координат траекторий t в текущем кадре.

Третий этап состоит из двух ассоциаций обнаружений и траекторий, для определения положения отслеживаемых объектов на текущем кадре. Для этого составляем матрицу близости между D_{high} и T на основе сходства по расстоянию IoU, внешнему сходству и согласованности направлений движения. Соответствия между траекториями и обнаружениями находим Венгерским алгоритмом, после чего добавляем в траекторию информацию об объекте для текущего кадра на основе координат обнаружения. Аналогично сопоставляем низкодостоверные обнаружения D_{low} и несопоставленные с первого этапа траектории $T_{re\text{-}remain}$. Несопоставленные траектории из $T_{re\text{-}remain}$ исключаем из списка отслеживаемых траекторий. Из несопоставленных высокодостоверных обнаружений генерируем новые траектории.

В случае офлайн-трекинга полученные траектории T проходят через дополнительный этап постобработки.

Алгоритм 1. *Базовый алгоритм трекинга*
Входные данные: Видеопоследовательность V; детектор для обнаружения объектов на кадрах Det; порог достоверности τ.
Выходные данные: Список траекторий T

```
T←∅
для кадров f_k в V:
    D_k ←Det(f_k)
    D_high← ∅
    D_low←∅
для обнаружений d в D_k:
    если d.score > τ:
    D_high←D_high ⋃ {d}
    иначе:
    D_low←D_low ⋃ {d}
для траекторий t в T:
    t←KalmanFilter(t)
    t←CameraMotionCompensation(t)
Ассоциация T и D_high, используя Венгерский алгоритм
D_remain ←несопоставленные обнаружения из D_high
T_remain ←несопоставленные траектории из T
Ассоциация T_remain и D_low, используя Венгерский алгоритм
T_re-remain ←несопоставленные траектории из T_remain
T←T \ T_re-remain
для обнаружений d в D_remain:
    T←T ⋃ {d}
конец
```

Обнаружение

Для обнаружения объектов используют детектирующую нейросеть YOLO v5 [18], которая позволяет осуществлять обнаружение и распознавание в реальном времени с достаточно высоким качеством (более 30 кадров в секунду). Практика показывает, что практически любой детектор не всегда может правильно классифицировать некоторые обнаруженные объекты, например, может возникнуть путаница при классификации между автомобилями и фургонами, мотоциклами и трициклами, людьми и пешеходами. Это приводит к инициализации нескольких отдельных траекторий для одного объекта и разрыву сопровождения объекта, так как стандартный алгоритм не отождествляет объекты разных классов в одну траекторию.

Чтобы разрешить эту проблему, предлагается сократить перечень классов, и при обработке данных детектора схожие классы объединять в один. Были

объединены в первый класс классы датасета человек и пешеход, во второй – автомобили, микроавтобусы и моторикши, в третий – велосипеды, мотоциклы, трициклы, в четвёртый – автобусы и грузовики. Затем к ним применяется алгоритм немаксимального подавления (non maximum supression, NMS [16]) для фильтрации множественных обнаружений одного объекта, отнесенного к разным классам. Конкретный класс в этом случае можно определить методом голосования, то есть как класс, который наиболее часто появляется в траектории данного объекта.

Модель движения

Для предсказаний местоположения на текущем кадре траекторий с прошлых кадров используется фильтр Калмана [13], инициализируемый для каждой траектории. Он содержит вектор состояния траектории $x_k=[u,v,s,r,u',v',s']^T$ на k-м кадре, где (u,v) – координаты центра на изображении, s – площадь, r – соотношение сторон, u', v', s' – соответствующие производные по времени. Цель алгоритма – предсказать новый вектор состояния $\hat{x_k}$ для кадра k на основе предыдущего вектора состояния x_{k-1}. Сам алгоритм состоит из двух этапов: прогнозирования вектора $\hat{x_k}$ и матрицы ковариации P_k и их корректировки, учитывая достоверную информацию о местоположении, полученной из сопоставленного этой траектории обнаружения после этапа ассоциации. На основе известного вектора состояния x_{k-1} и его матрицы ковариации P_{k-1} с прошлого кадра предсказывается вектор состояния $\hat{x_k}=Fx_{k-1}$, где F – матрица перехода между состояниями, и матрица ковариации $P_k'=FP_{k-1}F+Q$, где Q – матрица ковариации шума процесса. Корректировка предсказанного вектора состояния $\hat{x_k}$ осуществляется на основе измерения z_k, содержащего координаты обнаружения после ассоциации с траекторией. Новый вектор состояния x_k вычисляется по формуле вектора состояния $x_k=\hat{x}_k+K_k\left(z_k-H\hat{x}_k\right)$, где H – матрица для перевода вектора состояния в вектор наблюдения, K_k – коэффициент усиления Калмана, рассчитываемый по формуле $K_k=\hat{P}_kH^T\left(H\hat{P}_kH^T+R\right)^{-1}$, где R – шум наблюдений. И, наконец, корректировка матрицы ковариации вектора состояния $P_k=\left(I-K_kH\right)\hat{P}_k$.

Оригинальный фильтр Калмана предполагает, что объекты между двумя соседними кадрами по линейной модели, но это предположение часто оказывается неверно. Сами объекты могут двигаться по сложным траекториям, а движение камеры, установленной на беспилотном летательном аппарате (БЛА), добавляет еще большую нелинейность и непредсказуемость. Из-за этого новые координаты объектов могут предсказываться фильтром с большой погрешностью. Это приводит к тому, что ожидаемое положение объекта на текущем кадре очень сильно отличается от реального. Из-за этого объекты могут остаться не сопоставленными на этапе ассоциации, вследствие чего траектории объектов прерываются или теряются. Например, если в группе близко стоящих машин на рис. 1 начальное приближение (прогноз положения) неточно, алгоритм ByteTrack отождествляет машины по признаку дальности удаления от исходной точки некорректно.

Рис. 1. Результат трекинга объектов без модификаций

Для решения этой проблемы предлагается добавить в модель движения следующие элементы. Во-первых, добавить корректировку координат объектов из траекторий компенсированием движения камеры. Компенсация производится с помощью оценки матрицы трансформации W_k между кадрами k и $k-1$, и корректировки состояния фильтра Калмана $(u,v,1)^T=W(u,v,1)^T$ для каждого объекта. Матрицу W_k можно оценить методом RANSAC [23] по сопоставленным особым точкам из двух последовательных кадров, извлеченных с помощью дескриптора (ORB [21] или SIFT [22]) или использовать коэффициент корреляции последовательных изображений ECC для итеративного вычисления матрицы трансформации напрямую.

Во-вторых, предлагается использовать NSA модификацию фильтра Калмана [14]. Она заключается в том, чтобы добавить информацию о достоверности d_{score} обнаружений (оценке уверенности в наличии объекта, полученной детектором) в коэффициент усиления Калмана K_k на этапе обновления вектора состояния и матрицы его ковариации. В статье StrongSORT[6] предложено каждый элемент матрицы шума R наблюдения умножать на $(1-d_{score})$. Однако при таком подходе для обнаружений с низкой достоверностью также уменьшаем шум, но нередко сами обнаружения с низкой достоверностью значительно зашумлены. Поэтому вместо коэффициента $(1-d_{score})$ будем использовать коэффициент $((1-(d_{score}))$, где функцию f зададим как степенную $f_p(x)=x^p$. Экспериментально установлено значение $p=3$. Это позволяет влиять на матрицу шума обнаружений только обнаружениям с высокой достоверностью.

Влияние на качество работы фильтра Калмана продемонстрировано на рис. 2-4. Черным цветом показаны координаты из фильтра Калмана перед этапом ассоциации, белым – после этапа корректировки фильтра Калмана. Видно, что применение предложенных подходов позволяет фильтру Калмана предсказывать более точные местоположения на новых кадрах.

Ассоциация

На этапе ассоциации производится связывание уже отслеженных на предыдущих кадрах объектов,

представленных последовательностью своих координат на кадрах (то есть треков, траекторий) и обнаруженных на текущем кадре объектов (обнаружений). Далее множество изображений объектов на последовательных кадрах, сопоставленных одному реальному объекту, будем называть траекторией. Объект на текущем кадре, найденный детектором и пока не сопоставленный никакой траектории, будем называть обнаружением. Для выполнения связывания строится матрица близости траекториями и обнаружениями. По матрице близости с помощью Венгерского алгоритма строится оптимальный в смысле суммарного значения меры близости набор пар траекторий и обнаружений. В соответствии с полученным паросочетанием обнаружения привязываются к траекториям. Несопоставленные обнаружения инициализируют новые траектории.

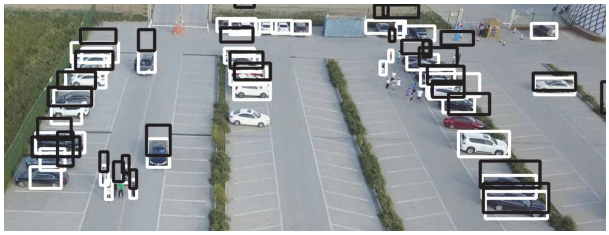

Рис. 2. Результат трекинга с применением модификации фильтра Калмана

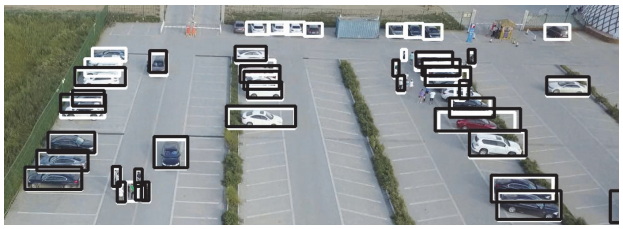

Рис. 3. Результат трекинга с применением компенсации движения камеры

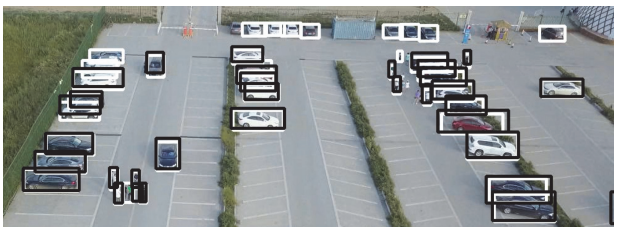

Рис. 4. Результат трекинга с применением компенсации движения камеры и модификации фильтра Калмана

Ключевым элементом алгоритма является функция вычисления меры сходства объектов. Традиционно мера сходства основывается на мере сходства ограничивающих прямоугольников обнаруженных объектов. В оригинальном ByteTrack матрица близости строится на основе меры Жаккара (Intersection over union, IoU) между ограничивающими прямоугольниками объектов на текущем и предыдущем кадре: $D_{IOU}=1-I_OU(i,j)$. Но она не дает возможность корректно оценить близость и сопоставить непересекающиеся объекты, так как при нулевом пересечении всегда даёт ноль даже совершенно разных комбинаций объектов. Чтобы устранить эту проблему, предлагается заменить меру IoU на обобщённую меру GIoU[24], т.е. $D_{IOU}(i,j)=1-(1+GI_OU(i,j))/2$. Для сопоставления объектов одного класса будем проводить ассоциацию между траекториями и обнаружениями, принадлежащими одному конкретному классу объектов, то есть для каждого класса будем производить свою итерацию ассоциации. Для более точной оценки сходства предлагается учитывать в матрице близости, помимо расстояния, сходство по внешнему виду объектов и согласованность направлений скоростей движения по предлагаемой ниже схеме.

Оценка сходства внешнего вида

Используемая линейная модель движения не всегда правильно предсказывает новое местоположение, вследствие чего может возникнуть ситуация, в которой модель движения неправильно предсказывает местоположение объекта, и в предсказанной области находится другой объект. В этом случае на основе меры расстояния произойдет ложное сопоставление. Поэтому имеет смысл учитывать внешний вид объектов для того, чтобы исключить сопоставление визуально различных объектов. Ряд алгоритмов используют для этого дескрипторы объектов, основанные на глубоком обучении. Однако их использование, во-первых, вычислительно затратно на этапе их извлечения, из-за чего их не получится быстро вычислять на центральном процессоре, а во-вторых, они имеют размерность до 512-1024, что значительно повысит время этапа ассоциации при вычислении сходства векторов и общее время работы алгоритма. Для решения этих двух проблем предлагается использовать вычислительно простой необучаемый дескриптор объектов, который строится как набор элементов (чисел или векторов), каждый из которых характеризует определённую характеристику внешнего вида объекта. Дескриптор вычисляется для объектов траекторий и для новых обнаружений. Затем производится их сравнение.

Для построения дескриптора внешнего вида $\boldsymbol{e}$ используются характеристики цвета объекта, характеристики размера объекта и характеристики яркости объекта. Для построения характеристики цвета объекта будем использовать гистограмму цветов с 5 бинами для каждого базисного цвета (красный, зелёный, синий). Для того чтобы учесть форму объекта, в дескриптор добавляется ширина и высота объекта. Распределение яркостей задаётся с помощью уменьшения изображения объекта до размера 3×3 и перевода в оттенки серого. Итоговый дескриптор имеет размерность 26 ($=5\cdot3+2+3\cdot3$) и получается конкатенацией из трех предыдущих. Он позволяет различать объекты по форме, распределению яркости и распределению цветов.

Помимо построения дескриптора, необходимо определить, какой дескриптор приписывать траектории после сопоставления. Можно заметить, что если

присваивать траектории дескриптор последнего сопоставленного обнаружения, то в случае частичной кратковременной окклюзии (перекрывания объекта другими) дескриптор траектории после прохождения окклюзии может очень сильно отличаться от дескриптора обнаружения этого объекта на последующих кадрах. Из-за этого могут отбрасываться верные сопоставления. Для решения такой проблемы хорошо зарекомендовал себя подход взвешенного усреднения дескрипторов (StrongSORT [6], BoT-SORT [19], GIAOTracker [14]), то есть использование экспоненциальной скользящей средней оценки для обновления дескриптора траектории $\tilde{e}$. Он обновляется по формуле $\tilde{e} = 0{,}9 \cdot \tilde{e} + 0{,}1 \cdot e$, где e – дескриптор обнаружения, который был сопоставлен траектории.

Сходство дескрипторов внешнего вида объектов и траекторий рассчитывается с помощью нормированного L1 расстояния межу дескрипторами:

$$D_{reid}(i,j) = k \frac{\widetilde{e_i} - e_j}{\widetilde{e_i}},$$

где $\widetilde{e_i}$ – сглаженный дескриптор i-й траектории, e_j – дескриптор j-го обнаружения, $\|\cdot\|$ – норма по L_1, k – эвристический гиперпараметр. Экспериментально установлено оптимальное значение $k = 3$.

Согласованность направления скоростей

Естественно предположить, что рассматриваемые объекты не могут изменить направление своей скорости мгновенно. Поэтому направления скорости будем считать ещё одним дескриптором объекта, который вычисляется перед этапом ассоциации. Для его расчёта за предыдущие n кадров определяется направление скорости объекта в траектории. Для этого вычисляются предполагаемые скорости за последние n кадров, то есть разница между координатами обнаружения на текущем кадре t и координатами объекта на траектории на кадре t-n. Координаты на предыдущих кадрах корректируются матрицами трансформации для минимизации влияния движения камеры на направление движения самого объекта. После чего между для i-й траектории и j-го обнаружения вычисляется скалярное произведение между векторами скоростей:

$$r(i,j) = \left(X^T_{t-n} - X^T_t\right)\left(X^T_{t-n+1} - X^O\right) + \\ + \left(Y^T_{t-n} - Y^T_t\right)\left(Y^T_{t-n+1} - Y^O\right),$$

где $(X^T_i,\ Y^T_i)$ – координаты центра объекта в траектории на i-м кадре, (X^O, Y^O) – координаты центра обнаружения. Затем в матрицу затрат добавляется нормированная на π оценка угла между векторами скоростей:

$$D_{velocity}(i,j) = v \cdot \arccos\left(r(i,j)\right) / \pi,$$

где v – гиперпараметр. Экспериментально установлено значение $v = 0{,}1$.

Это позволяет уменьшить вероятность сопоставления похожих объектов, движущихся в разные стороны.

Итоговая формула для вычисления значений в матрице близости между траекториями и обнаружениями состоит из меры сходства ограничивающих прямоугольников (с учётом коррекции движения камеры), меры сходства дескрипторов внешнего вида и меры сходства направлений скоростей:

$$D_{all}(i,j) = D(i,j)_{GIOU} \cdot D_{reid}(i,j) + D_{velocity}(i,j).$$

Постобработка

Этап постобработки нужен для корректировки результатов, полученных после онлайн-трекинга. В этой статье мы предлагаем улучшенную линейную интерполяцию пропущенных обнаружений и фильтрацию повторяющихся траекторий.

Интерполяция

Иногда при резком изменении движения или ракурса объект перестает отслеживаться на некоторое время. После того как потерянный объект снова стал отслеживаться, информация о его местоположении в моменты пропуска обнаружений отсутствует. Для заполнения пробелов информация о местоположении объекта на кадре интерполируется с учетом движения камеры следующим образом.

Предположим, что на кадре с индексом 0 объект с координатами c_0 был обнаружен и сопоставлен с траекторией, далее слежение прервалось и было восстановлено в положении c_{n+1} на кадре с индексом n. Интерполируем пропущенные координаты $\hat{c}_i$, где i – номер кадра, в котором объект не отслеживался.

Сначала предположим, что объект неподвижен. Обновленные координаты на i-м кадре c^*_i вычисляются рекуррентно: $c^*_1 = W_0 c_0$, $c^*_i = W_{i-1} c^*_{i-1}$ и

$$c^*_i = \left(\prod_{k=0}^{i-1} W_k\right) c_0,$$

где W_{k-1} – матрица перехода для координат от (k–1)-го кадра к k-у кадру, полученная из метода для компенсации движения камеры.

Затем учтем собственное движение объекта. Для этого оценим отклонение ожидаемого (при условии неподвижности) и реального положения объекта $c_{n+1} - c^*_{n+1}$ и выполним линейную интерполяцию его положения:

$$\hat{c}_i = c^*_i + \frac{c_{n+1} - c^*_{n+1}}{\mathrm{n}}.$$

Разницу применения алгоритма трекинга, описанного выше, и линейной интерполяции координат между позициями объекта в треке можно видеть на рис. 5-6.

Удаление дублирующихся траекторий

Дублирующиеся траектории часто возникают, когда детектор один и тот же объект на данном кадре ви-

деопоследовательности относит к разным классам, порождая два обнаружения вместо одного. Эти ошибки могут быть отсечены на этапе трекинга. Для этого удаляются траектории, которые в течение 30 % своего существования перекрывались траекторией, которая существует дольше, и имеют близкие дескрипторы внешнего вида своих объектов. Этот же алгоритм можно использовать для получения траектории из обнаружений, полученных разными детекторами на одном и том же кадре. Результат применения показан на рис. 7-8.

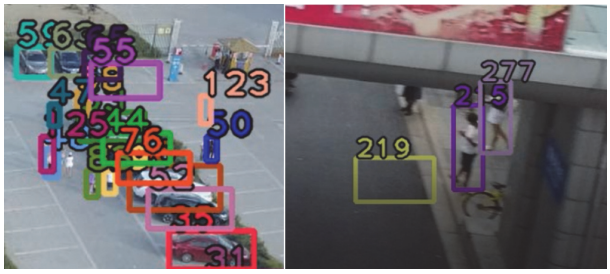

Рис. 5. Линейная интерполяция для предсказания положения объектов

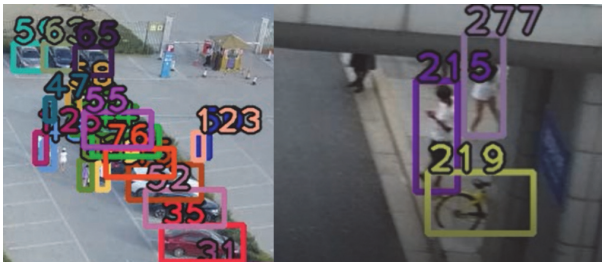

Рис. 6. Интерполяция положения объектов, устойчивая к движениям камеры

Рис. 7. Примеры дублирующихся траекторий объектов

Рис. 8. Результат применения фильтрации к дублирующимся траекториям

Результаты экспериментов

В ходе проведённых экспериментов было исследовано влияние различных модификаций и их комбинаций на результаты алгоритма трекинга на наборе данных Visdrone2019 MOT test-dev по метрикам IDF1 [29], MOTA [28], IDs [28]. Всего на тестовой части выборки присутствовало 17 видеопоследовательностей, 6635 кадров, 2461 уникальный объект. Дополнительно было проведено исследование устойчивости алгоритма трекинга к изменению частоты кадров (или, наоборот, скорости движения камеры и объектов). Для этого был сформирован разреженный вариант датасета, в котором был оставлен только каждый 4 кадр из видеопоследовательности.

Тестирование проводилось при реализации алгоритма на центральном процессоре Intel Core i5-10505 3.20 GHz.

Входными данными для алгоритма трекинга являлись результаты обнаружения детектора YOLO v5, который был обучен на изображениях разрешением 640×640, полученных из набора данных VisDrone2019-DET-train. Обнаружения разделялись по достоверности порогом 0,5. На этапе ассоциации отклонялись пары обнаружений и траекторий, если значение в матрице близости больше 0,8, траектории хранились на протяжении 30 последних кадров.

В табл. 1 приведено сравнение влияния трех различных методов компенсации движения камеры на итоговый результат трекинга. Обозначения следующие: orb – применение дескриптора ORB и метода сопоставления на основе RANSAC, sift – применение дескриптора SIFT и RANSAC, ecc – оценка смещения на основе коэффициента корреляции ECC. Видно, что наиболее эффективным является использование ECC, который будет использоваться в дальнейшем.

Табл. 1. Сравнение методов оценки межкадрового смещения

Метод	IDF↑	MOTA↑	IDs↓
orb	66,9 %	**49,2 %**	1465
sift	66,7 %	49,1 %	2255
ecc	**67,0 %**	**49,2 %**	**1448**

В табл. 2 представлено влияние на качество и скорость работы различных модификаций для алгоритма ByteTrack по отдельности. Обозначения следующие: class – применение группирования классов, nms – применение немаксимального подавления детекций (non max supression), reid – использование меры сходства внешнего вида объектов с помощью предложенного дескриптора, giou – замена меры IOU на GIOU, cmc – использование компенсации движения камеры, velocity – учёт согласованности направления скорости при оценке сходства объектов, nsa – учёт достоверности обнаружений в фильтре Калмана.

Табл. 2. Влияние различных модификаций на качество и время выполнения алгоритма трекинга

Алгоритм	IDF1↑	MOTA↑	IDs↓	Δtime↓, мс/кадр
Byte	62,7 %	48,7 %	2575	0
$\text{Byte}_{\text{class}}$	62,5 %	48,4 %	2542	+0,1
Byte_{nms}	62,9 %	48,7 %	2552	+1,2
Byte_{cmc}	66,1 %	49,4 %	1569	+0,9
$\text{Byte}_{\text{giou}}$	62,7 %	48,9 %	2205	+2,4
$\text{Byte}_{\text{reid}}$	63,8 %	48,8 %	2234	+8,6
$\text{Byte}_{\text{reid+giou}}$	65,5 %	49,3 %	1725	+11,7
Byte_{nsa}	63,3 %	48,9 %	2429	+0,3
$\text{Byte}_{\text{velocity+cmc}}$	66,5 %	**49,5 %**	1539	+1,5
$\text{Byte}_{\text{reid+giou+cmc}}$	**67,3 %**	**49,5 %**	**1393**	+15,2

Результаты показывают, что наиболее существенное влияние оказывают использование компенсации

движения камеры и использованного предложенной меры оценки сходства объектов.

В табл. 3 представлены результаты исследования различных вариантов комбинация модификаций для алгоритма ByteTrack на качество трекинга. Обозначение all – применение всех модификаций одновременно (all = class + nms + reid + giou + cmc + velocity + nsa). Результаты показывают, что наибольший интегральный прирост точности даёт совместное использование меры GIoU, дескриптора сходства внешнего вида и компенсация движения камеры. NSA и NMS в ряде случаев могут давать дополнительный прирост к качеству.

Табл. 3. Влияние различных комбинаций модификаций на качество

Алгоритм	IDF↑	MOTA↑	IDs↓
Byte	62,7%	48,7%	2575
$Byte_{class+nms}$	62,7%	48,7%	2560
$Byte_{class+nms+reid+giou}$	65,8%	49,3%	1648
$Byte_{class+nms+reid+giou+cmc}$	67,0%	49,2%	1448
$Byte_{class+nms+reid+giou+cmc+velocity}$	66,9%	49,2%	1462
$Byte_{class+nms+reid+giou+cmc+nsa}$	66,9%	49,2%	1451
$Byte_{reid+giou+cmc}$	**67,3 %**	49,5%	1393
$Byte_{nsa+reid+giou+cmc}$	**67,3 %**	**49,6 %**	1411
$Byte_{velocity+ giou+reid +cmc}$	**67,3 %**	**49,6 %**	1409
$Byte_{nms+ giou +reid+cmc}$	**67,3 %**	49,5%	**1370**
$Byte_{all}$	66,9%	49,2%	1456

В табл. 4 представлены результаты исследования качества работы различных версий алгоритма при уменьшении частоты кадров видеопоследовательности (FPS) в 4 раза. Обозначения аналогичны предыдущим таблицам. В этом случае межкадровые изменения слишком сильные, что создает трудности в работе немодифицированного фильтра Калмана, так как движения объектов становятся существенно менее предсказуемыми. Модификация NSA фильтра Калмана в этом случае справляется значительно лучше.

Табл. 4. Результаты при уменьшении частоты кадров в 4 раза

	IDF↑	MOTA↑	IDs↓
Byte	54,3%	42,2%	1816
$Byte_{nms+cmc+reid}$	63,9%	46,3%	1102
$Byte_{nms+reid+cmc+velocity}$	64,0%	46,4%	1104
$Byte_{nms+ reid + cmc + nsa}$	**64,3 %**	**46,8 %**	974
$Byte_{all}$	64,2%	46,6%	**962**

В табл. 5 представлено влияние предложенной постобработки на качество трекинга. В таблице Interpolation – применение предложенного алгоритма интерполяции, track_nms – применение алгоритма фильтрации дублирующих траекторий.

Эксперименты подтвердили, что замена меры IoU на GIoU, применение модулей NMS, оценки внешнего вида и компенсации движения камеры даёт наиболее значительный прирост качества. Это подтверждает эффективность этих модулей при решении задачи сопровождения объектов на данных, получаемых с борта БЛА, которые являются более сложными, чем данные от статичной камеры. Модификация NSA фильтра Калмана необходима в случае низкой частоты кадров видеопоследовательности. При этом учёт согласованности направления скоростей и классов объектов при сопоставлении не оказывают существенного влияния на качество.

Табл. 5. Влияние постобработки на качество трекинга

	IDF↑	MOTA↑	IDs↓
$Byte_{nms+cmc+reid}$	67,3 %	49,5 %	1370
$Byte_{interpolate}$	68,2 %	50,1 %	1089
$Byte_{interpolate+track\ nms}$	**68,2 %**	**50,0 %**	**1077**

Заключение

В данной статье была рассмотрена задача межкадрового сопоставления (трекинга) объектов на видеопоследовательностях. Рассмотрены методы повышения качества алгоритма трекинга для парадигмы трекинга на основе детектирования.

Были предложены модификации на каждом этапе для онлайн-отслеживания. На этапе обработки данных об обнаружениях: учёт класса объекта при сопоставлении, группирование классов и применение к ним немаксимального подавления для фильтрации множественных обнаружений объекта на изображениях. На этапе ассоциации объектов: компенсация движения камеры, учёт внешнего вида объекта и согласованности направления скоростей при вычислении матрицы близости, а также замена меры сходства областей IoU на GIoU. На этапе предсказания положений – учёт достоверности обнаружений в фильтре Калмана (NSA фильтр Калмана). Для офлайн-отслеживания предложены два метода для постобработки: интерполяция, устойчивая к движениям камеры, и фильтрация дублирующихся траекторий. Проведено экспериментальное исследование всех разработанных модификаций и их комбинаций.

Тестирование онлайн-отслеживания на тестовой части выборки Visdrone2019 показало, что за счёт предложенных модификаций мера качества IDF1 возросла с 62,3 % до 67,3 %, мера MOTA возросла с 48,7 до 49,5, IDs уменьшилось с 2575 до 1370. В результате офлайн-отслеживания IDF1 увеличился с 67,3 % до 68,2 %, MOTA – с 49,5 до 50,0 %, IDs уменьшилась с 1370 до 1077. Таким образом, предложенные модификации позволили существенно снизить долю случаев неверного связывания траекторий и новых обнаружений и, соответственно, повысить качество трекинга.

Разработанный алгоритм существенно превосходит базовый и позволяет уверенно решать задачу трекинга в сложных условиях при съёмке с нестационарной камеры, например, с бортовой камеры БЛА.

References

[1] Bewley A, Ge Z, Ott L, Ramos F, Upcroft B. Simple online and realtime tracking. 2016 IEEE Int Conf on Image Processing (ICIP) 2016: 3464-3468.

[2] Zhang Y, Sun P, Jiang Y, Yu D, Yuan Z, Luo P, Liu W, Wang X. Bytetrack: Multi-object tracking by associating every detection box. arXiv Preprint. 2021. Source: <https://arxiv.org/abs/2110.06864>.

[3] Wojke N, Bewley A, Paulus D. Simple online and realtime tracking with a deep association metric. 2017 IEEE Int Conf on Image Processing (ICIP) 2017: 3645-3649.

[4] Zhang Y, Wang C, Wang X, Zeng W, Liu W. FairMOT: On the fairness of detection and re-identification in multiple object tracking. Int J Comput Vis 2021; 129(11): 3069-3087.

[5] Stadler D, Beyerer J. Modelling ambiguous assignments for multi-person tracking in crowds. Proc IEEE/CVF Winter Conf on Applications of Computer Vision 2022: 133-142.

[6] Du Y, Song Y, Yang B, Zhao Y. StrongSORT: Make DeepSORT great again. arXiv Preprint. 2022. Source: <https://arxiv.org/abs/2202.13514>.

[7] Sun P, Jiang Y, Zhang R, Xie E, Cao J, Hu X, Kong T, Yuan Z, Wang C, Luo P. TransTrack: Multiple-object tracking with transformer. arXiv Preprint. 2020. Source: <https://arxiv.org/abs/2012.15460>.

[8] Zheng L, Tang M, Chen Y, Zhu G, Wang J, Lu H. Improving multiple object tracking with single object tracking. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition 2021: 2453-2462.

[9] Cao J, Pang J, Weng X, Khirodkar R, Kitani K. Observation-centric SORT: Rethinking SORT for robust multi-object tracking. arXiv Preprint. 2022. Source: <https://arxiv.org/abs/2203.14360>.

[10] Bergmann P, Meinhardt T, Leal-Taixe L. Tracking without bells and whistles. 2019 IEEE/CVF Int Conf on Computer Vision (ICCV) 2019: 941-951.

[11] Yan B, Jiang Y, Sun P, Wang D, Yuan Z, Luo P, Lu H. Towards grand unification of object tracking. European Conf on Computer Vision 2022: 733-751.

[12] Milan A, Leal-Taixe L, Reid I, Roth S, Schindler K. MOT16: A benchmark for multi-object tracking. arXiv Preprint. 2016. Source: <https://arxiv.org/abs/1603.00831>.

[13] Brown RG, Hwang PYC. Introduction to random signals and applied kalman filtering: with MATLAB exercises and solutions. 3rd ed. New York, NY: Wiley; 1997.

[14] Du Y, Wan J, Zhao Y, Zhang B, Tong Z, Dong J. GIAOTracker: A comprehensive framework for MCMOT with global information and optimizing strategies in VisDrone 2021. Proc IEEE/CVF Int Conf on Computer Vision 2021: 2809-2819.

[15] Evangelidis GD, Psarakis EZ. Parametric image alignment using enhanced correlation coefficient maximization. IEEE Trans Pattern Anal Mach Intell 2008; 30(10): 1858-1865.

[16] Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards real-time object detection with region proposal networks. NIPS'15: Proc 28th Int Conf on Neural Information Processing Systems 2015; 1: 91-99.

[17] Duan K, Bai S, Xie L, Qi H, Huang Q, Tian Q. CenterNet: Keypoint triplets for object detection. 2019 IEEE/CVF Int Conf on Computer Vision (ICCV) 2019: 6568-6577.

[18] Glenn J. YOLOv5 release v6.1. 2022. Source: <https://github.com/ultralytics/yolov5/releases/tag/v6.1>.

[19] Aharon N, Orfaig R, Bobrovsky B-Z. BoT-SORT: Robust associations multi-pedestrian tracking. arXiv Preprint. 2022. Source: <https://arxiv.org/abs/2206.14651>.

[20] Kuhn HW. The hungarian method for the assignment problem. Nav Res Logist Q 1955; 2(1-2): 83-97.

[21] Rublee E, Rabaud V, Konolige K, Bradski G. ORB: An efficient alternative to SIFT or SURF. 2011 Int Conf on Computer Vision 2011: 2564-2571.

[22] Lowe DG. Distinctive image features from scale-invariant keypoints. Int J Comput Vis 2004; 60(2): 91-110.

[23] Fischler MA, Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Comm ACM 1981; 24(6): 381-395.

[24] Rezatofighi H, Tsoi N, Gwak JY, Sadeghian A, Reid I, Savarese S. Generalized intersection over union: A metric and a loss for bounding box regression. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition (CVPR) 2019: 658-666.

[25] Bose B, Wang X, Grimson E. Multi-class object tracking algorithm that handles fragmentation and grouping. 2007 IEEE Conference on Computer Vision and Pattern Recognition 2007: .

[26] Shen Y, Hu W, Liu J, Yang M, Wei B, Chou CT. Efficient background subtraction for real-time tracking in embedded camera networks. SenSys '12: Proc 10th ACM Conf on Embedded Network Sensor Systems 2012: 295-308. DOI: 10.1145/2426656.2426686.

[27] Mahalanobis P. On the generalized distance in statistics. Proc Nat Inst Sci India 1936; 2: 49-55.

[28] Bernardin K, Stiefelhagen R. Evaluating multiple ob ject tracking performance: the clear mot metrics. EURASIP J Image Video Process 2008; 2008: 246309.

[29] Ristani E, Solera F, Zou R, Cucchiara R, Tomasi C. Performance measures and a data set for multi-target, multi camera tracking. European Conf on Computer Vision 2016: 17-35.

[30] Dendorfer P, et al. MOTChallenge: A benchmark for single-camera multiple target tracking. Int J Comput Vis 2021; 129(4): 845-881.

Сведения об авторах

Калугин Василий Федорович, студент бакалавриата НИУ «МЭИ», сотрудник лаборатории автоматического дешифрирования изображений ФАУ «ГосНИИАС». Область научных интересов – обнаружение и распознавание объектов на изображениях, трекинг объектов на видеопоследовательностях. E-mail: *v.f.kalugin@gmail.com* .

Горбачев Вадим Александрович, кандидат физико-математических наук, начальник лаборатории автоматического дешифрирования изображений ФАУ «ГосНИИАС». Окончил Московский физико-технический институт. Область научных интересов – компьютерное зрение, машинное обучение, нейросети, глубокое обучение.
E-mail: *vadim.gorbachev@gosniias.ru* .

ГРНТИ: 28.23.15
Поступила в редакцию 11 января 2023 г. Окончательный вариант – 18 июня 2023 г.

Development of a multi-object tracking algorithm with untrained features of object matching

V.A. Gorbachev [1], V.F. Kalugin [1]
[1] State Research Institute of Aviation Systems, 125167, Moscow, Russia, Viktorenko str. 7

Abstract

The problem of multiple object tracking is one of the most difficult tasks in computer vision. The article is devoted to a task of multiple object tracking on video footage received from an unmanned aerial vehicle. Unlike a static camera platform, the mobile platform causes an accidental camera movement, which leads to sudden changes in the position, angle and scale of objects. Such aspects considerably hinder efficient object tracking. In this paper, we explore the possibilities of improving the tracking quality in the case of camera movements. We significantly outperform ByteTrack algorithm, one of the best tracking algorithms for the MOT Challenge dataset, on the Visdrone 2019 dataset.

Keywords: multiple object tracking, YOLO v5, ByteTrack, Kalman filter, Visdrone 2019, UAV.

Citation: Gorbachev VA, Kalugin VF. Development of a multi-object tracking algorithm with untrained features of object matching. Computer Optics 2023; 47(6): 1002-1010. DOI: 10.18287/2412-6179-CO-1275.

Authors' information

Vasiliy Fedorovich Kalugin (b. 2001) student of the Moscow Power Engineering Institute in the direction of Applied Mathematics and Computer Science. Currently he works at the GosNIIAS in the laboratory of automatic image interpretation. Research interests are object tracking, object detection. E-mail: *v.f.kalugin@gmail.com* .

Vadim Alexandrovich Gorbachev, (b. 1988), PhD, head of automatic image interpretation laboratory in GosNIIAS. Research interests are object detection and tracking, neural networks, machine learning.
E-mail: *vadim.gorbachev@gosniias.ru* .

Received January 11, 2023. The final version – June 18, 2023.