ОБРАБОТКА ВИДЕОПОТОКА В СИСТЕМЕ ТЕХНИЧЕСКОГО ЗРЕНИЯ

Кузнецов А. П., Снисаренко С. В.
Кафедра систем управления,
Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: kafsu@bsuir.by

Представлен метод обнаружения объектов в зашумленном видеопотоке с высоким FPS в системе технического зрения с использованием библиотек: OpenCV для работы с изображениями и видео, NumPy для проведения операций над матричным представлением фреймов, Numba для ускорения части операций, которые выполняются в среде Python.

Введение

Системы технического зрения для обработки видеоинформации снабжаются необходимыми алгоритмами и программным обеспечением. Они могут иметь наборы прикладных программ реального времени для различных технологических задач с микропроцессорной обработкой информации. Тогда они чисто программным путем быстро переналаживаются для обслуживания тех или иных технологических операций. Следовательно, совместно с роботами, для очувствления которых системы технического зрения служат, и с другим оборудованием они представляют единый многофункциональный роботизированный технологический комплекс, способный автоматически работать в различных ситуациях как при смене вида операций, так и в не вполне определенной или меняющейся обстановке. Программное обеспечение в системах технического зрения зависит от алгоритмов обработки информации, т. е. от способов идентификации предметов и выбранных признаков для этого, а эти факторы существенно влияют на обеспечение требования работы системы в реальном времени. Детекция движения – актуальная проблема в решении задач видеоаналитики. Часто она осуществляется за счет сравнения изменяемой части изображения с неизменяемой, что позволяет выделить фон и движущийся объект.Точность выделения контура движущегося объекта зависит от точности выделения области движущегося объекта. Выделяют такие ошибки выделения области движущегося объекта, как шум, ложное обнаружение (ошибка первого рода) и пропуск цели (ошибка второго рода). Для решения данных проблем в основном применяют метод соответствующей фильтрации на стадии обнаружения.[1]

I. Постановка задачи

Для интеллектуальной системы обработки видеопотока, полученного с камеры технического зрения робота, необходимо реализовать программно алгоритм идентификации движения объекта, используя базовый метод определения движения. За базовый принят алгоритм вычитания фона с использованием смеси Гауссовых распределений. В основе базового алгоритма лежит усреднение, матричное вычитание и отсечение шума. Этапы алгоритма следующие:

1. Сжатие изображения для удаления лишних деталей с интерполяцией для лучшего усреднения;
2. Гауссовское размытие для удаления лишних деталей и снижения уровня зашумленности;
3. Вычитание из текущего фрейма фрейма фона; отсечение “тусклых” фрагментов, которые характерны для фона; выравнивание цвета “ярких” фрагментов, которые характерны для движения;
4. Поиск контуров движущихся объектов.

Данный алгоритм имеет два недостатка, которые должны быть учтены при программной реализации – влияние шума и изменяемость фона.Общая схема последовательности обработки видеопотока в интеллектуальной системе технического зрения (см. рис. 1).

Рис. 1 – Последовательность обработки видеопотока в системе технического зрения

После восприятия информации в виде визуального изображения производится ее предварительная обработка для снижения посторонних помех, улучшения изображений отдельных элементов обьекта или сцены, а затем она подвергается сегментации, заключающейся в подразделении сцены на составляющие части или элементы для выделения на изображении интересующих объектов. Последующее описание массива информации представляет собой определение характерных параметров, необходимых

для выделения требуемых объектов или элементов сцены и дальнейшего их распознавания посредством идентификации в соответствии с программным набором информации. И, наконец, посредством интерпретации окончательно устанавливается принадлежность "рассматриваемого"объекта к группе распознаваемых, установление его зрительного образа.

Алгоритм будет реализован с применением библиотеки компьютерного зрения OpenCV, которую часто применяют в роботах для распознавания объектов окружающего мира.

II. Программная реализация обработки видеопотока

При программной реализации алгоритма использовался следующий инструментарий Python: библиотека с открытым исходным кодом OpenCV для обработки видео, бибилиотека Numba для оптимизации ресурсоемких операций, бибилиотека NumPy для работы с матрицами. Рассмотрим этапы реализации.

Этап 1: Сжатие. На данном шаге происходит сжатие исходного изображения, что позволяет удалить мелкие детали и снизить уровень шума. Кроме того, это позволяет ускорить последующие операции и уменьшить количество потребляемой оперативной памяти.

Этап 2: Размытие. Уменьшенный фрейм подвергается гауссовскому размытию для того, чтобы еще более сгладить детали, выровнять фон, уменьшить шум и снизить влияние незначительных колебаний фона. Этап 3: Добавление в буфер движения. Циклический буфер движения используется для того, чтобы снизить уровень шума фона за счет усреднения. При этом в данном буфере новые кадры считаются более релевантными, чем старые. После того, как кадр вытесняется из буфера движения, он перемещается в циклический буфер фона. Размер буфера движения – от 3-5-10 фреймов (до 1/3 секунды). Чем больше кадров в буфере движения, тем выше вероятность обнаружения, но больше погрешность за счет включения в область движения “хвоста” объекта.

Этап 4: Добавление в буфер фона. Буфер фона – объемный буфер, в котором кадры формируют усредненную картину прошлого состояния среды за более продолжительное время, например, за 1 секунду для изменчивой среды, за 10 секунд для более стабильной среды. Чем больше кадров в истории буферного фона, тем меньший уровень шума создают краткосрочные изменения пространства, например, движение листвы, но дольше влияют объекты, которые продолжительное место находятся в кадре, например, высаживающая людей машина. Кадры, вытесняемые из буфера движения перемещаются в буфер фона, что позволяет задать “отставание” сцены фона от сцены движения.

Этап 5: Усреднение фона. Операция усреднения фона производится как среднее арифметическое среди кадров в буфере фона.

Этап 6: Усреднение движения. Операция усреднения движения производится в форме суммы кадров буфера движения с коэффициентами арифметической прогрессии, которая нормируется на сумму арифметической прогрессии. Это позволяет повысить значимость самых “свежих” кадров и понизить значимость старых кадров.

Этап 7: Вычитания фона из движения. На этом этапе производится определение мест, в которых фон отличается от анализируемого кадра, что соответствует движению. Результирующее изображение содержит ряд “цветных” пятен, которые соответствуют движущемуся предмету. На следующем изображении можно видеть результат данной операции (см. рис. 2).

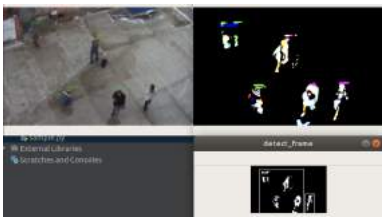

Рис. 2 – Удаление фона из движения

Наблюдается “хвост” у движущихся объектов – это побочный эффект повышения стабильности за счет усреднения. т.к. выполняется перевод в оттенки серого цвета. Далее, отсекаются “тусклые” части изображения, которые характеризуют или “хвосты” или зашумления. Отсечение производится по порогу белого цвета. Чем выше порог отсечения, тем большее количество движений будет проигнорировано – в частности, при движении объектов, цвет которых слабо отличается от цвета фона, они будут игнорироваться.

Этап 8: Уменьшение размера кадра движения. Алгоритм построения контуров объектов зависит от площади “белого” в кадре. Кроме того, на седьмом этапе объекты могут содержать изъяны для отображения, могут наблюдаться мелкие движения, которые должны игнорироваться, что может вести к детекции “шума”, ухудшению качества детекции, низкой производительности алгоритма. Для предотвращения этой ситуации необходимо дальнейшее уменьшение кадра.

Этап 9: Поиск областей, соответствующих движению и объединение пересекающихся областей. Алгоритм производит поиск в ширину по областям белого цвета, вычисляя охватывающие их рамки. Далее пересекающиеся рамки объединяются, чтобы уменьшить уровень дребезга.[2]

Рассмотрим особенности программной реализации. Чтобы обнаружить движение, необходимо вычислить разницу в значениях пикселей

двух изображений frame1 и frame2, предварительно их необходимо создать:

ret, frame1 = video.read()

ret, frame2 = video.read()

Для чтения видеопотока используется функция isOpened(), для этого организован следующий цикл:

while video.isOpened():

Далее необходимо вычислить абсолютную разницу между frame1 и frame2 с помощью методаcv2.absdiff():

difference = cv2.absdiff(frame1, frame2)

Далее необходимо перейти в оттенки серого с помощью метода cv2.cvtColor() т.к. обработка черно - белого изображения наиболее упрощена по сравнению с цветным и для идентификации движения объекта вполне применима:

gray = cv2.cvtColor(difference,

cv2.COLOR BGR2GRAY)

Согласно алгоритму, следующий шаг - это размытие изображения для удаления шума. Для этого используется метод cv2.GaussianBlur(). Он принимает несколько аргументов: исходное изображение для размытия, выходное изображение, размер ядра Гаусса, стандартное отклонение ядра по оси x, стандартное отклонение ядра по оси y и тип границы.[3]

blur = cv2.GaussianBlur(gray, (5,5), 0)

Затем необходимо установить пороговое значение с помощью метода cv2.threshold(). Данный прием выделит движение, сегментируя фон и передний план (или движение). Метод cv2.threshold() принимает четыре аргумента: изображение, пороговое значение, максимальное значение, и тип порогового значения.

threshold = cv2.threshold(blur,

20, 255, cv2.THRESH BINARY)

Далее используется метод cv2.dilate(), который принимает максимум 6 аргументов: изображение, ядро, привязку, итерации, тип границы и значение границы.

dilate = cv2.dilate (threshold, None, iterations=3)

Метод cv2.findContours() находит контуры объекта. Он принимает три аргумента: исходное изображение, режим поиска и метод аппроксимации контура.

contour= cv2.findContours(dilate, cv2.RETR TREE, v2.CHAIN APPROX IMPLE)

Метод cv2.drawContours() отрисовывает контуры объекта. Его аргументы: изображение, контуры, contourIdx (это значение отрицательно, если нарисованы все контуры), цвет, толщина, тип линии, иерархия, максимальный уровень и смещение.

cv2.drawContours(frame1, contour, -1, (0, 0, 255), 2)

Отображается изображение с помощью метода cv2.imshow().

cv2.imshow("image frame1)

Затем необходимо установить изображение frame2 в качестве первого кадра и считывать видео для нового кадра, который помещается в параметр frame2.[4]

frame1 = frame2

ret, frame2 = video.read()

Выход из цикла можно реализовать по нажатию заданной клавиши. Если клавиша «v» нажата, произойдет выход из цикла:

if cv2.waitKey(40) == ord('v'):

break

video.release().

Заключение

Шум и изменение фона в видеопотоке влекут за собой микроизменения в изображении, которые возникают даже в отсутствии явного движения в кадре, в следствие чего становится проблематичной обработка видеоизображения на предмет поиска движения и идентификации объекта. Базовые алгоритмы не обрабатывают данные эффекты. Алгоритм, рассматриваемый в данной статье позволяет успешно справляться с последствиями шума, а именно представлена программная реализация алгоритма обнаружения движения в видеопотоке системы технического зрения робота с подавлением возникших шумов и влияния изменнения фона. Использована библиотека алгоритмов компьютерного зрения и обработки изображений OpenCV для Python. Для увеличения скорости обработки видеопотока применена бибилиотека Numba, а для решении задач в рамках анализа данных с использованием матриц и векторов - библиотека NumPy.

III. Список литературы

1. Попов Е. П. Робототехника и гибкие производственные системы. / Е. П. Попов. - М.: Наука, 1987. - 190 с.
2. Высокопроизводительный детектор движения с подавлением шума на Python, OpenCV и Numba. [Электронный ресурс] / Bitworks software. – Томск, 2020. – Режим доступа: https://bitworks.software/high-speed-movement-detector-opencv-numba-numpy-python.html. – Дата доступа: 24.10.2021.
3. Шакирьянов Э. Д. Компьютерное зрение на Python. Первые шаги. /Э. Д. Шакирьянов. - Электрон. изд. - М. : Лаборатория знаний, 2021. - 163 с.
4. Fernández Villán, Alberto. Mastering OpenCV 4 with Python : A Practical Guide Covering Topics from Image Processing, Augmented Reality to Deep Learning with OpenCV 4 and Python 3. 7. / Fernández Villán, Alberto.- Packt Publishing Ltd, 2019. - 517 p.