Подсчет трафика

Баев А.Ж.

28 ноября 2018

Постановка

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (пример: часть размером 400×300)
цвет: RGB (по 1 байту на канал)
длительность: 2 часа (около 1 Гб)

Постановка

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (пример: часть размером 400×300)
цвет: RGB (по 1 байту на канал)
длительность: 2 часа (около 1 Гб)

Постановка

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (пример: часть размером 400×300)
цвет: RGB (по 1 байту на канал)
длительность: 2 часа (около 1 Гб)

Постановка

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (пример: часть размером 400×300)
цвет: RGB (по 1 байту на канал)
длительность: 2 часа (около 1 Гб)

Постановка

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (пример: часть размером 400×300)
цвет: RGB (по 1 байту на канал)
длительность: 2 часа (около 1 Гб)

Постановка

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (пример: часть размером 400×300)
цвет: RGB (по 1 байту на канал)
длительность: 2 часа (около 1 Гб)

Постановка

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (пример: часть размером 400×300)
цвет: RGB (по 1 байту на канал)
длительность: 2 часа (около 1 Гб)

Постановка

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (пример: часть размером 400×300)
цвет: RGB (по 1 байту на канал)
длительность: 2 часа (около 1 Гб)

Из пушки по воробьям

Библиотека opencv — очень мощный инструмент для задач компьютерного зрения реализованный на c++ и портированный на python.
Для пробного решения будем использовать его как открывашку и отображалку.

Ставим менеджер пакетов для python3. А через него уже ставим opencv.

```
sudo apt install python3-pip
pip3 install opencv-python
```

Узнать размер кадра

Таким образом можно открыть видеофайл и узнать размер кадра и глубину (каждый пиксель по умолчанию описывается 3 значениями — каналами цвета).

```
import cv2

cap = cv2.VideoCapture("file.mp4")
ret, frame = cap.read()
if ret:
    height, width, depth = frame.shape
    print(height, width)
```

Отобразить кадр на экране

Для отладки рекомендуется использовать вспомогательную графику. Для полноценного счета – её отключают.

```
import cv2

cap = cv2.VideoCapture("file.mp4")
ret, frame = cap.read()
cv2.imshow('Title', frame)
cv2.waitKey(0)
```

Если в качестве аргумент *waitkey* передать *n*, то изображение отображается *n* миллисекунд. Например, так *cv2.waitKey*(40) будет отрисовка 25 кадров в секунду.

Сохранить кадр как изображение

Для отладки иногда нужно анализировать отдельные кадры. Для этого лучше их сохранить в виде изображения.

```
import cv2

cap = cv2.VideoCapture("file.mp4")
ret, frame = cap.read()
cv2.imwrite("frame.jpg", frame);
```

Что такое frame

frame — это 3-мерный массив из библиотеки *numpy*.
первая координата — i от 0 до $height - 1$(номер строки сверху внизу),
вторая координата — j от 0 до $width - 1$ (номер столбца слева направо),
третья координата — k от 0 до 2 (цвета в порядка BGR),
значения — беззнаковое байтовое целое от 0 до 255.

Например так выглядит картинка 4 на 3 с 3 цветами.
$(0, 0, 0)$ — черный
$(255, 255, 255)$ — белый
$(0, 0, 255)$ — красный

```
[
[[0, 0, 0], [0, 0, 255], [255, 255, 100]],
[[20, 20, 20], [255, 0, 0], [255, 255, 100]],
[[100, 100, 100], [255, 0, 0], [255, 255, 100]],
[[100, 100, 100], [255, 0, 0], [255, 255, 255]]
]
```

Что такое frame

Массив можно редактировать прямым доступом. Проведем вертикальную красную линию по центру через весь кадр.

```
for i in range(height):
    frame[i, width // 2, 2] = 255
```

Лучше так не делать, а использовать функции *numpy* для генерирования массивов.

```
import numpy as np
frame[:, width // , 2] = 255 * np.ones(height)
```

Срезы numpy

Матрицы 3 на 3.

```
a = np.array([ [1, 2, 3, 4],
               [4, 5, 6, 7],
               [7, 8, 9, 10] ])
```

Срез подматрицы 2 на 2

```
b = a[0:2, 1:4]
```

Получим матрицу

```
2 3 4
5 6 7
```

Примитивы numpy

Вектор из 5 единиц

```
a = np.ones(5)
```

Матрицы 5 на 6 из чисел 7.0

```
a = np.ones((5, 6)) * 7.0
```

Матрицы 100 на 200 на 3 из чисел 0.0

```
a = np.zeros((5, 6, 3)) * 7.0
```

Диагональная матрица 5 на 5 с 2 по диагонали

```
a = np.eye(5)
```

Срезы

Если Вы что-то хотите сделать с непрерывным блоком матрицы, вероятнее всего вам нужны срезы (numpy-way), а не циклы.

Вырежем из большого кадра небольшой блок.

```
frame = frame[800:1200, 200:500]
```

Если такой код встречается часто, то лучше завести соответствующие объекты

```
xslice = slice(800, 1200)
yslice = slice(200, 500)
frame = frame[yslice, xslice]
```

Простое определение движения

1) Приводим исходные цветные данные к черно-белым (cvtColor) и к знаковому типу (иначе 100 - 200 = 155).

```
cap = cv2.VideoCapture("file.mp4")
xslice = slice(800, 1200)
yslice = slice(200, 500)

ret, img1 = cap.read()
src1 = img1[yslice, xslice]
frame1 = cv2.cvtColor(src1, cv2.COLOR_BGR2GRAY)
frame1 = frame1.astype(np.int8)

ret, img2 = cap.read()
src2 = img2[yslice, xslice]
frame2 = cv2.cvtColor(src2, cv2.COLOR_BGR2GRAY)
frame2 = frame2.astype(np.int8)
```

Простое определение движения

Всё кроме движущихся объектов в двух подряд идущих кадрах не меняется!

2) Вычисляем поэлементно модуль разности для матриц.

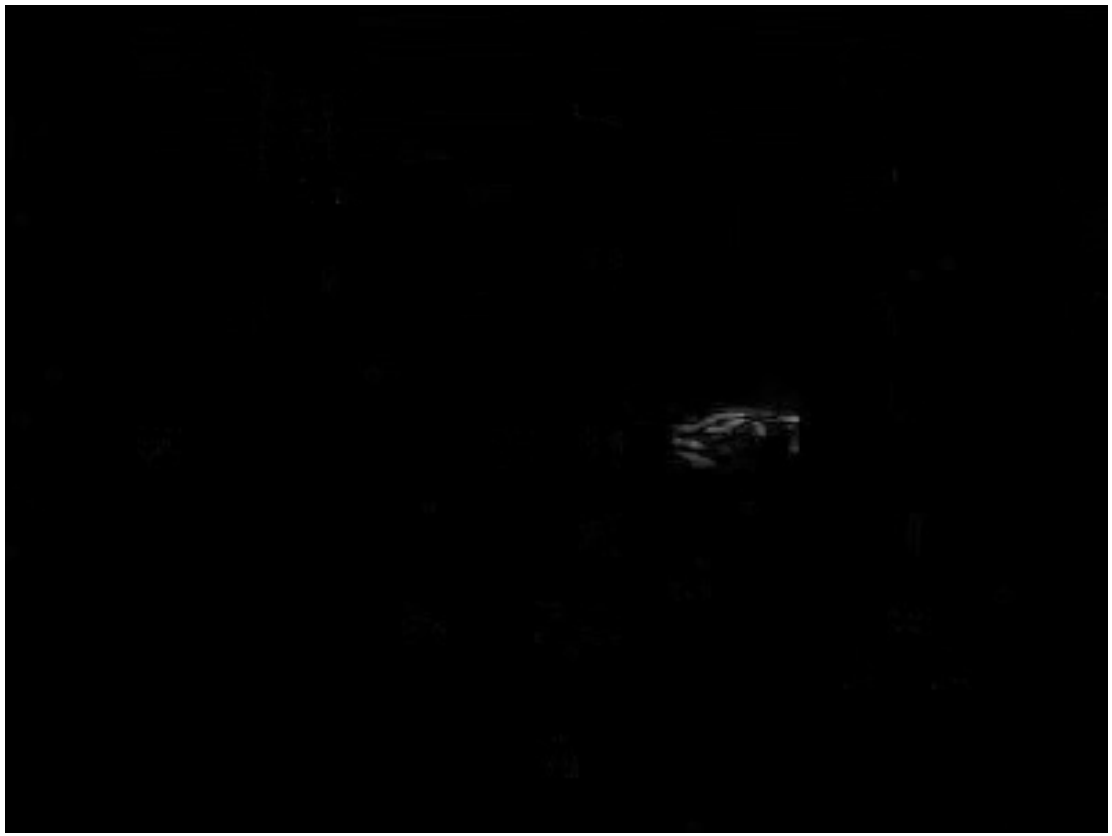
```
frame = np.abs(frame1 - frame2)
```

3) Приводим итоговые данные к беззнаковому типу.

```
frame = frame.astype(np.uint8)
```

Результат

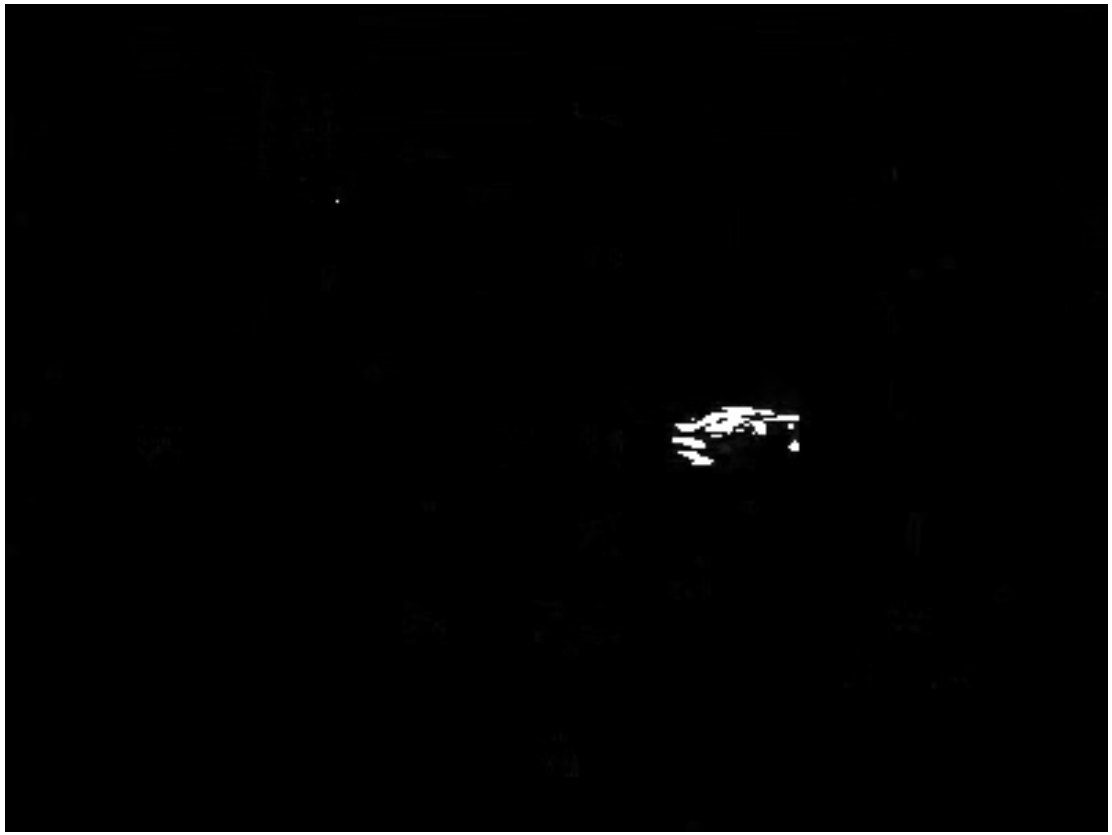
Не ярко?



Результат

Добавим контраста. Для этого применим изменения по маске

```
mask = np.zeros_like(frame)
np.putmask(mask, frame > 30, 255)
```



Результат

А еще лучше — наложим маску на исходные данные и раскрасим в хакерский цвет

```
out = src1
out[:,:,1] = np.maximum(out[:,:,1], mask)
```

Посмотрим анимацию

```
ret, img1 = cap.read()
src1 = img1[yslice, xslice]
frame1 = cv2.cvtColor(src1, cv2.COLOR_BGR2GRAY)
frame1 = frame1.astype(np.int8)

for i in range(10):
    ret, img2 = cap.read()
    src2 = img2[yslice, xslice]
    frame2 = cv2.cvtColor(src2, cv2.COLOR_BGR2GRAY)
    frame2 = frame2.astype(np.int8)

    frame = frame1 - frame2
    frame = np.abs(frame)
    mask = np.zeros_like(frame)
    np.putmask(mask, frame > 20, 255)

    out = src2
    out[:,:,1] = np.maximum(out[:,:,1], mask)
    cv2.imshow('frame' + str(i), out)
    cv2.waitKey(100)
    frame1 = frame2
```

Картинки

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (на примере область большого кадра размера 400×300)
цвет: черно-белый
длительность: 1 час

Картинки

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (на примере область большого кадра размера 400×300)
цвет: черно-белый
длительность: 1 час

Картинки

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (на примере область большого кадра размера 400×300)
цвет: черно-белый
длительность: 1 час

Картинки

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (на примере область большого кадра размера 400×300)
цвет: черно-белый
длительность: 1 час

Картинки

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (на примере область большого кадра размера 400×300)
цвет: черно-белый
длительность: 1 час

Картинки

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (на примере область большого кадра размера 400×300)
цвет: черно-белый
длительность: 1 час

Картинки

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (на примере область большого кадра размера 400×300)
цвет: черно-белый
длительность: 1 час

Картинки

Дан видео файл.
формат: .mp4
битрейт: 25 кадров в секунду
размер кадра: 1920×1280 (на примере область большого кадра размера 400×300)
цвет: черно-белый
длительность: 1 час

Идеи к алгоритму

Предварительная настройка

1. выбираем фиксирующее окно $W \times H$ (минимальный прямоугольник, в который попадают значительной частью все машины);
2. задаем чувствительность к движению в пикселе (по шкале от 0 до 255) — параметры маски;
3. задаем чувствительность к движению в окне (по шкале от 0 до $W \times H$) — минимальное количество изменяемых пикселей, которые в окне;

Обработка:

1. если в окне есть движение в течении B подряд идущих кадров, то фиксируем машину и блокируем окно на A следующих кадров

Сделайте лучше!

Текущие проблемы:

1. случайный помехи при шевелении камеры (ветер)
2. зависимость параметров от дня и ночи
3. ошибка пропуска — до 20% при плотном потоке
4. время расчета — сутки за 4 часа