Нижегородский государственный университет им. Н.И. Лобачевского
Институт информационных технологий, математики и механики
Кафедра Математического обеспечения и суперкомпьютерных технологий

Летняя школа ННГУ им. Н.И. Лобачевского по компьютерному зрению

Практическая работа №4
Сопровождение объектов с помощью решения задачи через построение траекторий их движения

Васильев Е.П.

Нижний Новгород
2022

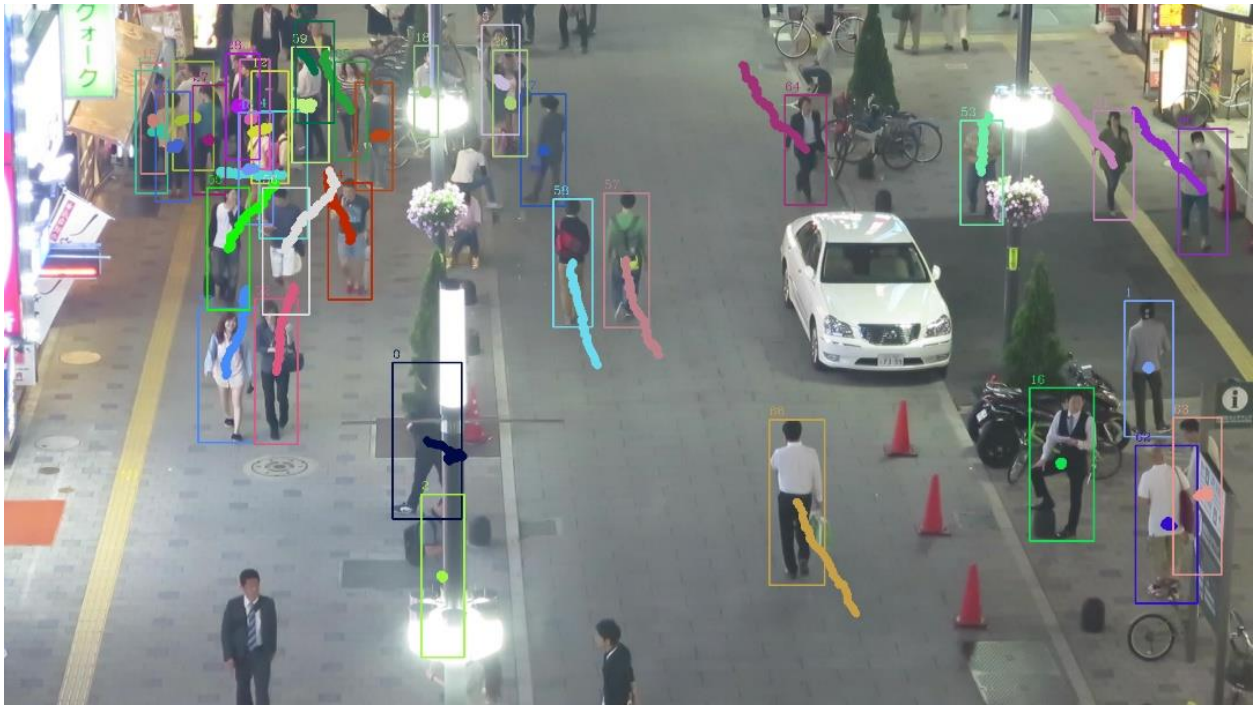
1 Цели и задачи работы

Цель работы состоит в решении задачи сопровождения (трекинга) объектов на видео через построение траекторий их движения

Для достижения поставленной цели необходимо решить следующие задачи:

- Изучить алгоритм сопровождения объектов через нахождение соответствий.
- Разработать приложение на базе компонента Inference Engine в составе OpenVINO для сопровождения объектов.

Отметим, что процедура настройки рабочего окружения подробно описана в предыдущей практике, поэтому данный шаг в настоящем описании опущен.



2 Алгоритм сопровождения объектов через нахождение соответствий

2.1 Общая схема алгоритма сопровождения

Алгоритм принимает на вход последовательность кадров видеопотока. На выходе формируется последовательность положений объектов интересующих классов на входной последовательности кадров. Формальная постановка задачи сопровождения объектов приведена в соответствующей лекции курса.

Общая схема алгоритма состоит из нескольких этапов. При этом предполагается, что на некотором шаге уже построены отдельные части траекторий для набора объектов, обнаруженных к текущему шагу.

1. Получить новый кадр видео.
2. Обнаружить объекты на полученном кадре.
3. Вычислить матрицу сходства между обнаруженными объектами и объектами, для которых построены отдельные части траекторий.
4. По матрице сходства ответить на следующие вопросы:
   4.1. Каким траекториям какой обнаруженный объект соответствует?
   4.2. Какие объекты появились впервые на видео (не соответствуют ни одной из уже существующих траекторий)?

4.3. Для каких траекторий на новом кадре не обнаружился объект (траектория завершилась в результате выхода объекта из области видимости камерой)?

5. В соответствии с полученными ответами обновить положения в траекториях, создать новые траектория для вновь обнаруженных объектов.
6. Повторить действия, начиная с шага 1, до момента завершения видео.

Первый шаг описанной схемы является техническим и выполняется стандартными средствами библиотек. Второй шаг представляет собой решение задачи детектирования объектов интересующих классов на изображении. Решение данной задачи с использованием методов глубокого обучения рассматривалось подробно в предыдущей практической работе. Здесь слушателю предлагается воспользоваться имеющимися знаниями и навыками. Рассмотрим детально шаги 3 – 5.

2.2 Вычисление матрицы сходства

Матрица сходства A для N траекторий и M объектов – это матрица размера $N \times M$, где каждый элемент a_{ij} представляет собой коэффициент сходства траектории T_i с объектом R_j. По данной матрице можно найти наилучшее соответствие между обнаруженными объектами и существующими траекториями.

Самый простой способ вычисления коэффициента сходства траектории T_i и нового объекта R_j выглядит следующим образом:

1. Получить последнее положение объекта в траектории T_i.
2. Сравнить объект, накрываемый окаймляющим прямоугольником в последнем положении траектории, и обнаруженный объект R_j по некоторому признаку.

Для сравнения можно использовать один или несколько следующих признаков: расположение, форму и внешний вид объекта (размер). Пусть D – расстояние между центрами окаймляющих прямоугольников (лежит на пересечении диагоналей), (w_1, h_1) – ширина и высота последнего прямоугольника в траектории, (w_2, h_2) – ширина и высота обнаруженного прямоугольника, C_1, C_2 – веса, определяющие вклад признака в результирующий коэффициент сходства. Тогда формула для вычисления коэффициента сходства по положению имеет вид:

$$affinity\_position = e^{-C_1\left(\frac{D^2}{w_1 h_1}\right)},$$

а формула для вычисления коэффициента сходства по размеру –

$$affinity\_shapes = e^{-C_2\left(\frac{w_1 - w_2}{w_1} + \frac{h_1 - h_2}{h_1}\right)}.$$

Исходя из этого, результирующий коэффициент сходства определяется следующим образом:

$$a_{ij} = affinity\_position * affinity\_shapes.$$

2.3 Поиск соответствий по матрице сходства

Задача поиска соответствий траекторий и обнаруженных положений по матрице сходства сводится к задаче о назначениях [**Ошибка! Источник ссылки не найден.**]. В канонической формулировке задача о назначениях имеет вид:

1. Имеется некоторое число работ $\{1, \dots, N\}$ и то же самое число исполнителей $\{1, \dots, N\}$.
2. Любой исполнитель i может быть назначен на выполнение любой (но только одной) работы $j = f(i)$, с затратами $a(i, j) \geq 0$.
3. Нужно распределить работы так, чтобы выполнить работы с минимальными суммарными затратами, т.е. необходимо минимизировать функционал $\sum_j a\big(i, f(i)\big)$.

Задача поиска наилучших соответствий траекторий и положений по матрице сходства – задача максимизации суммарного сходства. Чтобы свести нашу задачу к задаче о назначениях, необходимо выполнить следующие действия:

1. Сделать матрицу A квадратной. Для этого можно добавить некоторое количество пустых строк и столбцов, заполненных нулями.
2. Перейти от задачи максимизации к задаче минимизации. Поскольку $0 \leq a_{ij} \leq 1$, то достаточно заменить каждый элемент в матрице сходства по формуле $a'_{ij} = 1 - a_{ij}$.

Для решения полученной задачи можно воспользоваться функцией **`linear_sum_assignment`** пакета **`scipy.optimize`**, которая реализует Венгерский алгоритм [**Ошибка! Источник ссылки не найден.**, **Ошибка! Источник ссылки не найден.**].

2.4 Обновление набора траекторий

После того как найдены соответствия траекторий и вновь обнаруженных объектов, остается обновить множество траекторий.

1. Если коэффициент сходства между объектом и траекторией превышает пороговое значение, то добавить объект в траекторию.
2. Если объект не добавлен ни в одну из имеющихся траекторий, то необходимо создать новую траекторию и добавить в нее обнаруженный объект.

3 Решение задачи сопровождения на видео из набора данных MOT

В директории 4 практики расположен набор файлов, необходимых для решения задачи сопровождения на видео из набора данных MOT. Данное видео разбито на кадры, и для каждого кадра имеется набор обнаруженных на нем объектов (поскольку вычисление объектов каждый раз очень затратно).

Данный набор файлов полностью реализует алгоритм сопровождения, кроме трех методов класса **`Tracker`** (файл **`tracker.py`**):

- **`_calc_affinity_appearance()`**
- **`_calc_affinity_position()`**
- **`_calc_affinity_shape()`**

Данные методы необходимо реализовать самостоятельно, чтобы получить результаты трекинга.

При реализации данных функций, чтобы получить окаймляющие прямоугольники для траекторий и для нового объекта, используются следующие конструкции:

```
track.last().bbox
obj.bbox
```

Также в файлах папки common присутствуют полезные функции для подсчета площади bbox и т.д.

В файле annotation.txt для каждого обнаруженного объекта представлены координаты ограничивающего прямоугольника и вектор-описание внешнего вида (которые можно использовать в алгоритме сопровождения для повышения качества).

Чтобы запустить процедуру сопровождения, скачайте архив с данными (https://dropmefiles.com/9DzeY)

Данный архив имеет следующую структуру:

```
├── разметка
│   └── annotation.txt - файл разметки для запуска приложения отслеживания
└── MOT17-04-FRCNN
├── gt
│   └── gt.txt
└── img1 ---- папка с изображениями
├── 000001.jpg
├── 000002.jpg
│  ...
├── 001049.jpg
└── 001050.jpg
```

Чтобы запустить процедуру сопровождения, распакуйте архив и выполните команду

```
python main.py --annotation <uncompressed_dir>\annotation\annotation.txt --dst_folder .\DST1\
```

Если вы корректно реализуете функции сравнения, то вы увидите высокий Recall.

```
python main.py --annotation <uncompressed_dir>\annotation\annotation.txt --dst_folder .\DST1\
INFO|main.py:47|main: Begin reading annotation from the file 'annotation\annotation.txt' and emulating reallife detector issues
INFO|main.py:52|main: End reading annotation from the file 'annotation\annotation.txt' and emulating reallife detector issues
INFO|main.py:61|main: Begin tracking
Tracking:
100%|████████████████████████████████████████████████████████████████████████████████|
100/100 [00:01<00:00, 65.36it/s]
INFO|main.py:71|main: End tracking
INFO|main.py:73|main: Begin converting tracks and writing to file
Converting:
100%|████████████████████████████████████████████████████████████████████████████████|
34/34 [00:00<00:00, 72.96it/s]
INFO|main.py:79|main: End converting tracks and writing to file
INFO|main.py:81|main: Begin evaluation
INFO|main.py:84|main: Num groundtruth tracks that were found: num_hits=34
INFO|main.py:85|main: Num groundtruth tracks that were missed: num_misses=0
INFO|main.py:86|main: Total quality: recall=0.97
INFO|main.py:88|main: End evaluation, the result is written to the file DST1\evaluation.txt
INFO|main.py:90|main: Begin writing demo to images in the folder DST1
Writing images:
100%|████████████████████████████████████████████████████████████████████████████████|
100/100 [00:03<00:00, 26.08it/s]
INFO|main.py:92|main: End writing demo to images in the folder DST1
INFO|main.py:93|main: Done
```

Чтобы для ускорения запустить процедуру сопровождения для первых 100 кадров, распакуйте архив и выполните команду

```
python main.py --annotation <uncompressed_dir>\annotation\annotation.txt --dst_folder .\DST1\ --max_frame_index 100
```

После этого в папке **DST1** появятся файлы с результатами работы алгоритма сопровождения в текстовом формате. Чтобы увидеть результаты сопровождения на изображениях, добавьте параметр **--images_folder**, тогда в папке DST1 появятся изображения с нарисованными траекториями. Такой вариант использования работает медленнее.

```
python main.py --annotation annotation\annotation.txt --dst_folder .\DST1\ --images_folder MOT17-04-FRCNN\img1\
```

С дополнительным параметром --show будет происходить вывод изображений на экран.

```
python main.py --annotation annotation\annotation.txt --dst_folder .\DST1\ --images_folder MOT17-04-FRCNN\img1\ --show
```

4 Дополнительные задания

Реализуйте сохранение результатов сопровождения не в качестве изображений, а в виде видео (модифицируйте файл **demonstrator.py**).

5 Литература

1. Шолле Ф. Глубокое обучение на Python. – СПб.: Питер. – 2018. – 400с.
2. Girshick R., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. – 2014.
3. Dai J., Li Y., He K., Sun J. R-FCN: Object detection via region-based fully convolutional networks. – 2016.
4. Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.Y., Berg A.C. SSD: Single Shot MultiBox Detector. – 2016.
5. Redmon J., Divvala S., Girshick R., Farhadi A. You only look once: Unified, real-time object detection. – 2015.
6. Рамальо Л. Python. К вершинам мастерства / Пер. с англ. Слинкин А.А. – М.: ДМК Пресс. – 2016. – 768 с.
7. Страница репозитория Open Model Zoo [https://github.com/openvinotoolkit/open_model_zoo].
8. Документация Intel Distribution of OpenVINO Toolkit [https://docs.openvino.ai/latest/index.html].