Multi-object tracking на основе библиотеки Norfair

Сичкар Георгий

Tracking

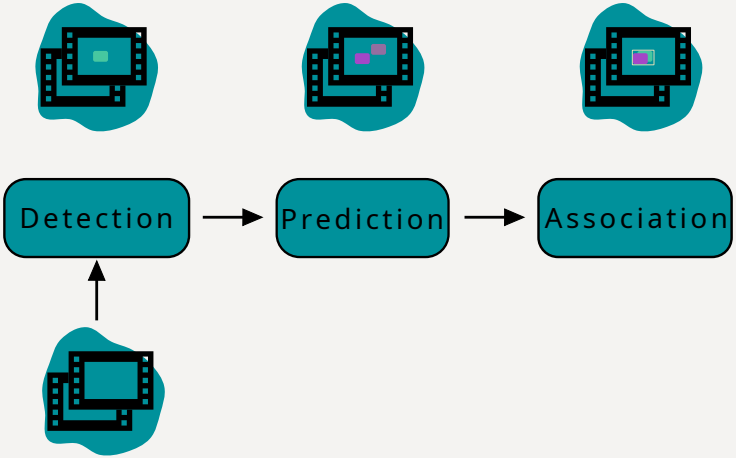
Object tracking —это задача отслеживания объекта на протяжении всего видео.

- Single object tracking (SOT) —обнаружение объекта не требуется (видеонаблюдение, видеотрансляции)
- Multi-object tracking (MOT) —одним из этапов является детектирование объектов (робототехника, автономное движение)

Где применяется

- Спортивный анализ — отслеживания мяча в теннисе, волейболе, футболе (Hawk-Eye Innovations)
- Видеонаблюдение — отслеживание необычной активности (HIKVISION)
- Робототехника — сложные взаимодействия (с человеком)

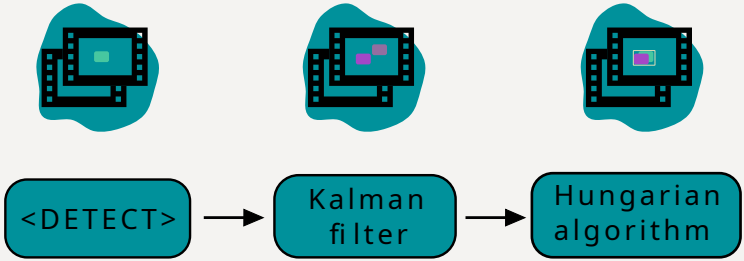
Этапы трэкинга



Существующие алгоритмы

- SORT — использует Kalman-фильтр и Венгерский алгоритм
- DeepSORT — расширение SORT (дополнительно использует сверточную нейронную сеть, для определения идентичности объекта)
- ByteTrack
- FairMOT
- GOTURN
- ...

SORT (идейно)



Существующие реализации с открытым исходным кодом

- OpenCV
- AlphPose (используется для трэкинга людей)
- dblib (только SOT)
- Ultralytics YOLO (алгоритм трэкинга — ByteTrack, YOLO для детекции)
- Norfair
- ...

Norfair

Norfair

Norfair — это настраиваемая облегченная библиотека Python для отслеживания нескольких объектов в реальном времени.

- Пользовательский детектор
- Алгоритм трэкинга — SORT
- Обработка видео — OpenCV
- Лицензия — BSD 3-Clause 😱🤯😳

MOT с norfair

```python
from norfair import Tracker, Video

video = Video(input_path="my_video.mp4")
tracker = Tracker()
detector = MyDetector()

for frame in video:
    detections = detector(frame)
    tracked_objects = tracker.update(detections=detections)
    draw_tracked_objects(frame, tracked_objects)
    video.write(frame)
```

MOT с norfair

```
from norfair import Tracker, Video

video = Video(input_path="my_video.mp4")
tracker = Tracker()
detector = MyDetector()

for frame in video:
    detections = detector(frame)
    tracked_objects = tracker.update(detections=detections)
    draw_tracked_objects(frame, tracked_objects)
    video.write(frame)
```

Norfair

```python
from norfair import Tracker, Video

video = Video(input_path="my_video.mp4")
tracker = Tracker()
detector = MyDetector()

for frame in video:
    detections = detector(frame)
    tracked_objects = tracker.update(detections=detections)
    draw_tracked_objects(frame, tracked_objects)
    video.write(frame)
```

Как создать MyDetector

```python
from norfair import Detection

class IDetector(Protocol):
    @abstractmethod
    def __call__(self, frame: np.ndarray) -> Iterable[Detection]:
        pass

class MyDetector(IDetector):
    def __call__(self, frame: np.ndarray) -> Iterable[Detection]:
        object_points: numpy.ndarray
        ...
        return [Detection(point) for point in object_points]
```

Как создать MyDetector (*IDetector*)

```python
from norfair import Detection
class IDetector(Protocol):
    @abstractmethod
    def __call__(self, frame: np.ndarray) -> Iterable[Detection]:
        pass

class MyDetector(IDetector):
    def __call__(self, frame: np.ndarray) -> Iterable[Detection]:
        object_points: numpy.ndarray
        ...
        return [Detection(point) for point in object_points]
```

Как создать MyDetector

```python
from norfair import Detection

class Detector(Protocol):
    @abstractmethod
    def __call__(self, frame: np.ndarray) -> Iterable[Detection]:
        pass
class MyDetector(Detector):
    def __call__(self, frame: np.ndarray) -> Iterable[Detection]:
        obj_points: np.ndarray
        ...
        return [Detection(point) for point in obj_points]
```

Detection

- `points` — набор точек, описывающих объект трекинга
- `scores` — уверенность детектора в том, что он правильно распознал объект

```python
from norfair import Detection

detection = Detection(
    points: np.ndarray,
    scores: np.ndarray
)
```

TrackedObject

- `estimate` — где будут точки объекта в текущем кадре
- `id` — «имя объекта»
- `age` — сколько фрэймов пережил
- `last_detection` — последнее связывание с Detection

```python
from norfair import TrackedObject

tracked_objects: list[TrackedObject]
tracked_objects = tracker.update(...)
```

Tracker

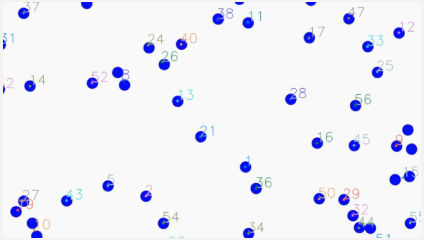
```python
from norfair import Tracker

tracker = Tracker(
    distance_function: str | Callable,
    distance_threshold: float,
    hit_counter_max: int,
    detection_threshold: float,
    filter_factory: FilterFactory
)
```

Tracker

- `distance_function` — используемая трекером для определения расстояния между `TrackedObject` и `Detection`
- `distance_threshold` — максимальное расстояние, которое может считаться совпадением между `TrackedObject` и `Detection`
- `hit_counter_max` — сколько подряд случившихся «совпадений» могут влиять на срок жизни `TrackedObject`
- `detection_threshold` — пороговое значение (если `score` у точки ниже, то точка отбрасывается)
- `filter_factory` — может быть использована для изменения параметров фильтра (KalmanFilter)

Примеры

полное виде



полное виде

Дополнительные материалы

- Ссылка на norfair
- Статья про рыб (с norfair)

Ссылка с примерами использования norfair!!!