Краткое описание

Система видео аналитики, способная распознавать, сопровождать, классифицировать и предсказывать траектории движения до 200 объектов одновременно. Каждый объект снабжается своим уникальным id, который не теряется, когда одновременно движутся более 200 объектов. Система работает быстрее чем реальное время благодаря оболочке pytorch детектора объектов, способной использовать все имеющиеся видеокарты и работать в несколько потоков.

Детекция объектов происходит на базе YOLO darknet либо SSD, обученных на базе собственной разметки, использующей как реальные, так и синтетические данные. Синтетические данные формировались на базе 3D моделирования

Предсказание траектории происходит на базе LSTM. Классификация поведения объектов происходит на базе алгоритмов, использующих методы линейной алгебры. Индивидуальный дескриптор каждого объекта, позволяющий одновременно отслеживать много объектов в условиях их временного исчезновения или затемнения, строится на базе технологии triplet loss, реализованной в pytorch.

Библиотеки

Модуль os - предоставляет множество функций для работы с операционной системой, причём их поведение, как правило, не зависит от ОС, поэтому программы остаются переносимыми. Здесь будут приведены наиболее часто используемые из них.

Numpy - библиотека с открытым исходным кодом для языка программирования Python. Возможности: поддержка многомерных массивов; поддержка высокоуровневых математических функций, предназначенных для работы с многомерными массивами

Модуль Random - предоставляет функции для генерации случайных чисел, букв, случайного выбора элементов последовательности.

Matplotlib - библиотека на языке программирования [Python](https://ru.wikipedia.org/wiki/Python) для визуализации данных [двумерной (2D) графикой](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0#%D0%94%D0%B2%D1%83%D0%BC%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0_(2D)) ([3D графика](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0#%D0%A2%D1%80%D1%91%D1%85%D0%BC%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0) также поддерживается). Получаемые изображения могут быть использованы в качестве иллюстраций в публикациях

SciPy - библиотека Python с открытым исходным кодом, предназначенная для решения научных и математических проблем. Она построена на базе NumPy и позволяет управлять данными, а также визуализировать их с помощью разных высокоуровневых команд.

Time - модуль для работы со временем в Python

Sklearn - библиотека машинного обучения для языка программирования Python.

PIL - библиотека языка Python, предназначенная для работы с растровой графикой

Torchvision – Библиотека для компьютерного зрения

PyTorch - фреймворк машинного обучения для языка Python с открытым исходным кодом, созданный на базе Torch. Используется для решения различных задач: компьютерное зрение, обработка естественного языка.

Основная часть

Для начала рассмотрим главный файл, где происходит работа трекера. Это файл facenet.ipynb. Формат файла это jupyter notebook - среда разработки, где сразу можно видеть результат выполнения кода и его отдельных фрагментов. Отличие от традиционной среды разработки в том, что код можно разбить на куски и выполнять их в произвольном порядке. В этом файле мы выбираем данные и параметры для обработки:

1. Должен определиться тип входных данных. Есть 4 варианта выбора:
   1. Видео. Флаг - 0
   2. Папка с нарезанными кадрами из видео. Флаг - 1
   3. RTSP-поток или поток с веб-камеры. Флаг - 2
   4. UDP-поток или поток данных с другой машины. Флаг - 3

RTSP-поток и UDP-поток работают в режиме реального времени. Для первого или второго варианта надо подавать уже готовые данные. После того как определились с типом входных данных, в переменной input\_type\_flag нужно поставить определенный флаг.

1. Если выбирается видео или папка с кадрами из видео, то надо подать путь до видео. Переменная: path\_in. Если надо получать картинку в режиме реального времени, то ставим нужный флаг и никакой путь подавать не надо.
2. Данный трекер работает на разных видеоадаптерах, таких как CPU/GPU. Для выбора адаптера есть переменная: video\_adapter. Работает также по флагу: 0 - CPU, 1 – GPU.
3. Одним из важных элементов является сам Трекер. Для его работы используется предиктор detector\_facenet\_pytorch, основанный на машинном обучение на базе pytorch и библиотека Face Recognition - автоматическая локализация лица на фотографии и, в случае необходимости, идентификация персоны по лицу. На каждом кадре, который подается в трекер, векторизуются все лица, находящиеся на изображении. Каждое лицо представляет собой вектор размера 512. Для того, чтобы идентифицировать лицо каждого человека ему дается уникальный id. Чтобы не потерять и не перепутать человека, каждый новый кадр сравнивает новый вектор с предыдущим, который находится в истории, и по косинусному расстоянию между векторами находит самый близкий к нему.
4. Для визуального представления, где в пространстве находятся векторизованные лица, был реализован механизм, использующий диффузную карту. Для этого есть специальная переменная show\_diffusion. 0 - выключена диффузная карта, 1 - включена на запись. Фрагменты сохраняются в папку “output”.
5. Используя трекер и откалибровав камеру, мы можем получать относительные координаты в пространстве каждого человека. Для визуального представления существует переменная show3D\_one\_points. 0 - не будет сохранять и рисовать визуализацию, 1 - будет рисоваться только 1 точка головы. Все нарисованные фрагменты лежат в папке “save\_fig”.
6. Также к трекеру добавлен и скелет человека. Это реализовано с помощью библиотеки mediapipe holistic. В этом скелете имеется 33 уникальных точки тела человека, включая лицо. Скелет ещё не реализован до конца, поэтому включать и выключать его можно по флагу, переменная add\_holistic. 0 - выключить скелет, 1 - включить скелет. Этот скелет можно представить на 3д карте, чтобы визуально наблюдать как это выглядит. Для этого есть переменная show\_3Dpoints\_holistic. 0 - не будет рисовать, 1 - Будет рисоваться скелет. Все фрагменты лежат в папке “save\_fig”.
7. Для визуального представления, как обрабатывается каждый кадр, у каждого человека рисуются специальные прямоугольные и подписываются id каждого человека. Если прямоугольник синий, то объект детекции распознан, если красный, то либо не понимает, либо объект потерялся в кадре. Специальная переменная show\_BBOX – включает и выключает отрисовку превью по флагу. 0 – выключена отрисовка, 1 – включена.
8. Для качественной работы трекера у каждого распознанного объекта лица есть специальный параметр “confidences”, или достоверность. Для того, чтобы убрать ложные детекции есть специальные порог в виде переменной thr\_confidence. Все детекции меньше порога удаляются.
9. Также существует Лист параметров, где вводятся переменные для работы трекера:
   1. 'quality\_width' – Выбор качества выходного изображения. Берется только ширина изображения, высота берется их сохранения пропорций первого кадра.
   2. 'check\_period' – каждый кадр равный этой переменной вызывает функцию forget\_horizont\_vert, в которой обрезается ненужное прошлое.
   3. l\_forget\_history' – количество истории, которое необходимо иметь для обработки текущего кадра.

Кроме основного файла есть используемые. Они находятся в папке z\_utils, и в них находятся функции, обеспечивающие работу трекера. Самым основным из них является файл Tracker.py. В нем вызываются все основные функции, начиная от инициализации параметром, заканчивая сохранением результатов. Основу его работы опишем ниже:

1. TrackObjects000=init\_000(flag\_prediction) - объекты инициализируются центрами прямоугольных объектов, распознанных нейросетью. Объектам присваивается класс первого распознавания
2. Инициализируем предиктор нейросети - detector\_facenet\_pytorch,
3. Распознавание объектов.
   1. imageIn = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB) – Считывание текущего кадра
   2. XXX, frame1, encodings = facenet\_pytorch\_application:
      1. XXX - создается структура прямоугольных детекций головы, его ширина, высота и центр на изображении в пикселях.
      2. frame1 – картинка в виде тензора
      3. encodings – каждая прямоугольная детекция головы описывается вектором размером 512.
   3. pts = object2pts(TrackObjects000.object\_) - получаем центры каждого лица и его айди
   4. q01, status, error = cv2.calcOpticalFlowPyrLK(old\_gray, gray\_frame, pts['coords'], None, \*\*lk\_params) – получаем сдвиг точек на следующий фрейм алгоритмом cv2. Optical Flow
   5. TrackObjects000.predict\_03() - предсказываем траектории
   6. q0=correct\_opt\_flow\_02(q01, TrackObjects000.predict ,1,thr\_correct\_opt\_flow) - если значения Optical Flow отличаются от предсказания на величину большую чем thr\_correct\_opt\_flow, то сдвигаем точки на базе предсказания
4. Для каждого объекта ищем подходящую коробку в окрестности сдвинутой точки при помощи OptFlow. она или есть, или ее нет. Корректируем объекты исходя из коробок порожденных сетью, дополняем историю. Если коробка для объекта не найдена, то объект считается не подтвержденным , и точка объекта сдвигается в соответствие с OpticflFlow (отображается на изображении квадратиком) в противном случае координата, полученная OFlow, корректируется центром соответствующего ей ббокса (объект отображается кружочком)
5. TrackObjects000.Update\_04(pts,collect\_color\_featres\_k,pts\_from\_cnn,thr\_find\_near,thr\_find\_near\_begin,thr\_find\_near\_big,thr\_color\_features,thr\_measure\_hue\_optflow2predict) – одна из главных функций в трекере. Сравнивает векторизованное лицо с предыдущего кадра и текущим. По наименьшему косинусному расстоянию между векторизованными лицами присваивает id на текущем кадре. У каждого объекта есть свой радиус, поэтому каждый объект сравнивает косинусные расстояния только с теми объектами, которые попадают в его радиус детекции. Переменная радиуса называется manual\_thr\_radius. Если объект векторизован, то рисуется синий прямоугольник. Если объект потерялся, то рисуется красный.
6. TrackObjects.Forget\_horizont\_vert(l\_forget\_history) - забываем объекты, которые долго не подтверждались нейросетью, либо если они подошли к краю изображения. Обрезаем историю в соответствие с установленной максимальной длиной, переменная: 'l\_forget\_history'.
7. TrackObjects.add\_new\_object - Если появился некий объект из структуры нейросети XXX, который не находится ни в одном из известных объектов, а также далек от центров известных объектов на расстоянии более manual\_thr\_radius, то добавляем его в трекер , присваиваем ему следующий по счету id.
8. TrackObjects.classyfy\_all\_trajectory() - классифицируем траектории. Перебираются все объекты трекера. Каждый объект снабжен траекторией - историей движения. На основе обучения эта траектория локально классифицируется по последним 10 отсчетам: 1 или -1. Если траектория достаточно длинная, проводится голосование, чего больше в течение трека, 1 или -1. В первом случае объект объявляется положительным и рисуется красным (см фигуры внизу), во втором – отрицательным (синее). Есть еще нейтральное состояние (белое), когда среднее показание класса близко к 0.
9. TrackObjects.Draw(frame1, self.frame\_counter) – функция предназначена для отрисовки прямоугольников у каждого лица и подписи его id. Также при включении трекера в работу, даёт превью картинку для визуального представления происходящего.
10. TrackObjects.cut\_and\_save\_objects - вырезаем картинку и сохраняем в полноценное видео, когда трекер закончит работу.
11. diffuse\_map.vizu\_with\_labels – функция для создания фрагментов диффузной карты. Каждое векторизованное лицо преобразуется из вектора размера 512 в трехкоординатное представление в пространстве. Каждый кадр сохраняется в отдельную папку для того, чтобы отследить насколько близко в пространстве находятся векторизованные лица.

Также есть скелетизация, которая находится в тестовом формате. Это реализуется с помощью двух функций в трекере:

1. TrackObjects.holistic\_model(frame1) – На вход получает текущий кадр. Так как это модель может обрабатывать только одного человека одновременно. Для этого каждого человека, которого векторизовал трекер, мы вырезаем и загружаем в эту модель. Из этой модели мы получаем относительные 33 точки описывающие скелет человека.
2. TrackObjects.transate\_landmarks\_to\_3d – специальная функция для перевода относительных точек скелета в пространстве в реальные 3D точки относительно камеры.

У этой модели скелета есть минус в виде медленной скорости обработки.