С. В. Снисаренко

ДЕТЕКТОР ОБЪЕКТОВ В РЕАЛЬНОМ ВРЕМЕНИ В СИСТЕМЕ ТЕХНИЧЕСКОГО ЗРЕНИЯ

Представлен метод обнаружения объекта в видеопотоке и измерения расстояния до него в режиме реального времени в системе технического зрения с помощью библиотеки OpenCV и технологии RealSense с использованием сверточной нейронной сети SSD

Введение

В промышленных роботах самое важное – это скорость, плавность и точность движений, которая должна повторяться тысячи раз. Производители промышленных роботов предлагают роботов с частично открытым кодом. Часть кода, которая отвечает за безопасность движения, полностью закрыта, а часть кода, отвечающая за сопряжение с искусственным интеллектом и управление роем роботов – открыта. Это упрощает интеграцию логистических роботов в существующие бизнес процессы и системы управления складом или производством. Это позволяет разработчикам производить всевозможные аппликации, решения, аксессуары и расширения для роботов под требуемые операции. Робот в реальном времени распознает объекты, корректирует свою траекторию, производит перестроение зон безопасности и т.д. Такие методы управления позволяют все больше снижать участие человека в производственном процессе. В ходе внедрения роботов в производство отечественные и зарубежные интеграторы используют целый ряд технологий – обработка изображений, ориентация в пространстве, облачные технологии и пр. Применение роботов на конвейерном производстве для перемещения заготовок требует высокой точности их позиционирования. Это не всегда возможно и тогда на помощь приходит техническое зрение. Цифровая камера получает изображение заготовки в рабочей зоне робота, ПО его анализирует, формулирует перед роботом задачу и тот ее выполняет. Задачи, которые можно решать при помощи машинного зрения: контроль процесса сборки изделия, подсчет объектов, измерение их параметров и многие другие. Задача обнаружения объектов на изображении сегодня является одной из ведущих в области машинного зрения. Ее суть заключается в том, чтобы не только классифицировать объект на снимке, но и указать его точное местоположение. Результаты обнаружения объекта могут быть дополнены информацией о том, насколько далеко расположен данный объект. В данной работе представлен метод решения задачи распознавания объекта и измерения расстояния до него в режиме реального времени с помощью библиотеки OpenCV и технологии RealSense.[1]

I. Постановка задачи

Необходимо реализовать программно детектор распознавания объектов в реальном времени, используя концепцию глубокого обучения (Deep Learning – DL) и библиотеку алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом OpenCV, чтобы работать с видеопотоком, получаемым с камеры технического зрения промышленного робота.

Задача измерения расстояния до объекта решается с помощью камеры глубины Intel RealSense D435, измеряющей глубину в каждой точке.

II. Программная реализация детектора объектов в реальном времени

При программной реализации детектора объектов в реальном времени использовался высокоэффективный класс VideoStream библиотеки OpenCV, поддерживающий многопоточность и высокую скорость обработки FPS(«frames per second»).[2] Обобщенный алгоритм реализации следующий.

1. Подключение пакетов и настройка параметров. Для программной реализации детектора объектов в реальном времени с помощью OpenCV необходимо получить доступ к камере технического зрения робота и непосредственно к видеопотоку. Далее необходимо применить распознавание объекта для каждого кадра. Для реализации поставленной задачи на первом этапе необходимо импортировать следующие модули: imutils из классов VideoStream и FPS для первичной обработки видеопотока; NumPy - для работы с матрицами, Argparse – для обработки аргументов командной строки; Time – для работы со временем; cv2 – для основных операций с изображениями. Для работы с командной строкой необходимо задать следующие аргументы: -prototxt – путь к prototxt Caffe файлу с информацией о нейронной сети и ее обучению; model – путь к предварительно подготовленной модели, confidence – минимальный порог валидности (сходства) для распознавания объекта (значение по умолчанию – 20%).

2. Добавление основных объектов и инициализация списка необходимых классов и набора цветов.

CLASSES = ["background . . .]

COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

Инициализируются метки CLASS и соответствующие случайные цвета.

3. Загрузка модели нейронной сети и настройка видеопотока. Загружается сериализованная модель, предоставляя ссылки на prototxt и модель.

net = cv2.dnn.readNetFromCaffe

(args["prototxt"],args["model"])

Затем инициализируется видеопоток с камеры технического зрения. Сначала запускается поток VideoStream, затем - ожидание включения камеры, начинается отсчёт кадров в секунду.

vs = VideoStream(src=0).start()

time.sleep(2.0)

fps = FPS().start()

Классы VideoStream и FPS являются частью пакета imutils.

4. Прохождение по каждому кадру. Для увеличения скорости, если порог валидности низкий, то кадры можно пропускать. Считывание кадра из видеопотока, выбор размера, получение ширины и высоты фрейма:

while True:

frame = vs.read()

2frame = imutils.resize(frame, width=400)

(h, w) = frame.shape[:2]

blob = cv2.dnn.blobFromImage

(cv2.resize(frame, (300, 300)),

0.00533, (300, 300), 125)

Передача фрейма как входных данных в заданную нейросеть и распознавание объектов, заданных в CLASSES.

net.setInput(blob)

detections = net.forward()

5. Фильтрация объектов. Допустим, обнаружены объекты в видеопотоке. Необходимо оценить значение валидности для отрисовки рамки вокруг объекта и его идентификации. Распознавание объекта и получение значения валидности:
for i in np.arange(0, detections.shape[2]):

confidence = detections[0, 0, i, 2]

Если значение валидности выше заданного порога, извлекаем индекс метки в классе и рассчитываем координаты рамки вокруг обнаруженного объекта.

if confidence > args["confidence"]:

idx = int(detections[0, 0, i, 1]) , box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])

Затем извлекаем (x;y)-координаты рамки, которые будем использовать для отображения прямоугольника и текста.

(startX, startY, endX, endY) = box.astype("int")

Нанесение текстового лейбла, содержащего имя из CLASS и значение валидности: . label = ": :.2f".format(CLASSES[idx],confidence * 100)

Отрисовка цветного прямоугольника вокруг объекта, используя цвета класса и раннее извлечённые (x;y)-координаты:

cv2.rectangle(frame,(startX, startY),(endX, endY), COLORS[idx], 2)

y = startY-15 if startY - 15>15 else startY+15

cv2.putText(frame, label, (startX, y),

cv2.FONTHERSHEYSIMPLEX, 0.5, COLORS[idx], 2)

6. Отображение кадра, проверка ключа выхода, обновление счётчика FPS:

cv2.imshow("Frame frame)

key = cv2.waitKey(1)&0xFF

if key3==ord("q"): break

fps.update()

Остановка счётчика FPS:

fps.stop()

Закрытие видеопотока:

cv2.destroyAllWindows()

vs.stop().

Детектор обнаруживает объекты разного размера согласно заданному набору цветов и датасету в режиме реального времени.

III. Измерение расстояния до объектов

Результаты обнаружения объекта могут быть дополнены информацией о том, насколько далеко расположен данный объект. Задачу измерения расстояния можно решить с помощью камеры глубины Intel RealSense D435, измеряющей глубину в каждой точке.[3] Измеряем расстояние до объекта:

depth=depthimage[int(left):int(right),

-int(top):int(bottom)].astype(float)

depth_scale=profile.getdevice()

.firstdepthsensor().get_depthscale()

depth=depth * depth_scale

dist,,, = cv.mean(depth)

dist = round(dist,1)

cv.putText(colorimg,"dist:"+str(dist)+"m

(int(left),

int(top)-5),cv.FONTHERSHEYPLAIN,

1,(0,255,0),1)

Таким образом можно оценить расстояние до выделенного объекта в пределах четырех метров, если периметр рабочей ячейки робота больше заданного значения, алгоритм будет иметь погрешность. Для увеличения точности замера можно применить к карте глубин дополнительные фильтры, заложенные в pyrealsense2, увеличивающие качество изображения, либо модифицировать сам алгоритм вычисления глубины (напр. вычислять взвешенное среднее или замерять расстояние в одной центральной области).[3]

IV. Использование сверточной нейронной сети

Глубокое обучение является мощным методом машинного обучения, который автоматиче-

ски изучает функции изображений, требуемые для задач обнаружения. Существует несколько методов для обнаружения объектов с помощью глубокого обучения, таких как Faster R-CNN, YOLO, SSD. Архитектура сверточной сети(см. рис. 1).

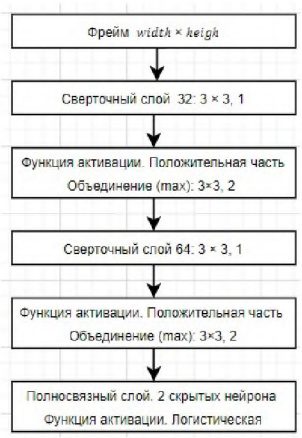

Рис. 1 – Архитектура сверточной нейронной сети

Модель Single Shot Detector(SSD) использует идею применения пирамидальной иерархии выходов свёрточной сети для эффективного обнаружения объектов различных размеров. Изображение последовательно передаётся на слои свёрточной сети, которые уменьшаются в размерах. Выход из последнего слоя каждой размерности участвует в принятии решения по детекции объектов. Это позволяет обнаруживать объекты различных масштабов. SSD не разбивает изображение на сетку произвольного размера, а предсказывает смещение ключевых рамок с масштабированием, одна размерность выходного слоя отвечает за объекты своего масштаба. В результате, большие объекты могут быть обнаружены только на более высоком уровне, а маленькие объекты — на низких уровнях. На этапе обучения реализована возможность идентифицировать образцы, до этого система уже обучена на dataset. Параметры, включенные в расчет, называются «весами», функция расчета отклонения от метки называется «функцией потерь», коэффициент компенсации потерь называется «смещением». На этапе прогнозирования (распознавания) система считывает данные выборки, передает их через набор вычислений и сравнивает их с обученными данными и назначается метке, которая имеет лучшую точность в наборе данных, независимо от того, обучена ли она. [4]

Для полноценной работы детектора объектов в реальном времени необходимо иметь предварительно подготовленную Convolutional Neural Network.

python realtimeobjectdetection.py
prototxt MobileNetSSDdeploy.prototxt.txt
model MobileNetSSDdeploy.caffemodel

В данной работе использована реализация SSD TensorFlow Пола Баланса, доступная на GitHub.[5] Выявлены следующие недостатки данной сверточной нейронной сети: SSD путает объекты с похожими категориями из CLASSES, возможно, из-за наличия идентичных атрибутов.SSD дает худшую производительность для небольших объектов, так как они могут детектироваться не на всех кадрах видеопотока. Увеличение разрешения входного изображения облегчает эту проблему, но не решает ее полностью.

Заключение

В работе рассмотрена задача обнаружения и идентификации множества объектов в видеопотоке, а также задача измерения расстояния до объекта. Данный метод предлагает использовать сверточную нейронную сеть(SSD) на каждом кадре видеопотока для детекции объектов, заданных в списке классов с заданием цветов на языке Python с использованием библиотеки для технического зрения и машинного обучения OpenCV. Используя значение порога валидности объекта в видеопотоке можно изменять скорость работы детектора, а также выделять и наносить метки на объект. Задача измерения расстояния решена с помощью камеры глубины Intel RealSense D435, измеряющей глубину в каждой точке.

V. Список литературы

1. Технологические тенденции развития промышленных роботов. [Электронный ресурс] / TAdviser.Государство. Бизнес. Технологии , 2020. – Режим доступа: `www.tadviser.ru/index.php/` Статья:Технологические тенденции развития промышленных роботов – Дата доступа: 24.03.2022.
2. Samyak Datta. Learning OpenCV 3 Application Development.Published by Packt Publishing Ltd, 2016. - 305 p.
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.
5. Github.com. [Электронный ресурс] / Product.Code, 2017. –Режим доступа:`https://github.com/balancap/SSD-Tensorflow/Code`. – SSD:balancap/SSD-Tensorflow – Дата доступа: 28.03.2022.

Снисаренко Светлана Валерьевна, аспирант кафедры систем управления БГУИР, snisarenko@bsuir.by.

Научный руководитель: Кузнецов Александр Петрович, доктор технических наук, профессор, kuznap@bsuir.by.