МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по производственной практике

Тема: Моделирование движения дрона в задаче слежения за движущимся объектом

Студент гр. 8383 _____ Киреев К.А.

Руководитель _____ Ульянов С.А.

Санкт-Петербург

2021

ЗАДАНИЕ

НА ПРОИЗВОДСТВЕННУЮ ПРАКТИКУ

Студент Киреев К.А.

Группа 8383

Тема практики: Моделирование движения дрона в задаче слежения за движущимся объектом

Задание на практику:

Выполнить численное моделирование движения квадрокоптера или автономного подводного аппарата (далее, робота) в задаче слежения за движущимся объектом по видеоизображениям с использованием библиотеки `PyBullet`.

Сроки прохождения практики: 30.06.2020 – 20.07.2021

Дата сдачи отчета: 16.07.2021

Дата защиты отчета: 16.07.2021

Студент _____ Киреев К.А.

Руководитель _____ Ульянов С.А.

АННОТАЦИЯ

Задача слежения состоит в удержании некоторого движущегося объекта с известными формами и габаритами в центре видеоизображения. В идеале от робота также требуется хотя бы приблизительно удерживать расстояние до объекта. Считаем, что камера закреплена и не изменяет своего положения относительно корпуса. Для начала также считаем, что в среде нет других объектов и препятствий

SUMMARY

The tracking task is to keep some moving object with known shapes and dimensions in the center of the video image. Ideally, the robot is also required to maintain at least an approximate distance to the object. We assume that the camera is fixed and does not change its position relative to the body. To begin with, we also assume that there are no other objects and obstacles in the environment.

СОДЕРЖАНИЕ

1. РЕШЕНИЕ ЗАДАЧИ

1.1. Обработка изображений камеры

Для получения и последующего рендеринга изображения объекта на сцене, по сути, надо сначала разместить виртуальную камеру и направить ее на объект. `PyBullet` делает снимки моделирования довольно простыми с помощью функции `getCameraImage`, которая визуализирует `RGB`-изображение, маску сегментации и буфер глубины одновременно. Однако нужно сначала указать свойства камеры, такие как физическое местоположение камеры, целевую точку просмотра, ориентацию камеры, `FOV`, разрешение и расстояние рендеринга объекта. Снимки моделирования представлены на рис. 1.

Листинг 1.

```
viewMatrix = pb.computeViewMatrix(
    cameraEyePosition=[0+(horz_shift / 128), 0+(vert_shift / 128), 5],
    cameraTargetPosition=[0+(horz_shift / 128), 0+(vert_shift / 128), 0],
    cameraUpVector=[0, 1, 0])

projectionMatrix = pb.computeProjectionMatrixFOV(
    fov=35.0,
    aspect=1.0,
    nearVal=0.1,
    farVal=5.1)

width, height, rgbImg, depthImg, segImg = pb.getCameraImage(
    width=128,
    height=128,
    viewMatrix=viewMatrix,
    projectionMatrix=projectionMatrix)
```

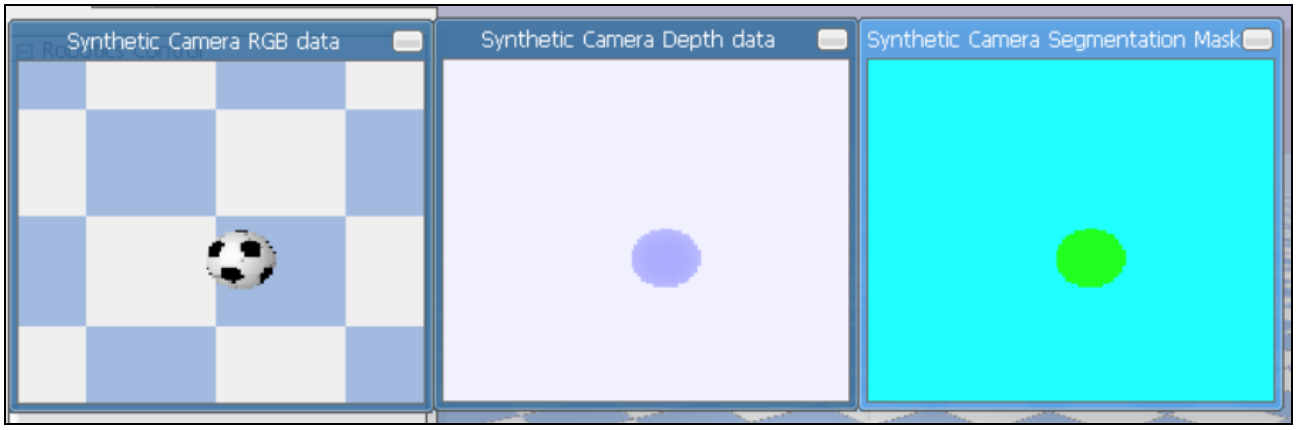

Рисунок 1 – Снимки моделирования

Одним из важных направлений искусственного интеллекта является компьютерное зрение. Компьютерное зрение — это наука о компьютерах и системах программного обеспечения, которые могут распознавать и понимать изображения и сцены. Компьютерное зрение также состоит из различных аспектов, таких как распознавание изображений, обнаружение объектов, генерация изображений, супер-разрешение изображений и многое другое.

Для решения поставленной задачи слежения за объектом необходимо приметь подход к распознаванию объекта на изображении. Было принято решение использовать `ImageAI` - библиотеку `python`, которая позволяет легко интегрировать технологии компьютерного зрения в приложение. Была использована модель `YoloV3`, которая будет использоваться для обнаружения объектов на изображении. Функция `detectObjectsFromImage` обрабатывает `RGB`-изображение и возвращает несколько параметров, таких как название, вероятность предсказания. Также она возвращает координаты нижнего левого и верхнего правого углов рамки объекта, которые и будут нужны в дальнейшем для перемещения дрона. Обработанное изображение представлено на рис. 2.

Листинг 2.

```
detector = ObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath("models/yolo.h5")
detector.loadModel()

def detect():
    detections = detector.detectObjectsFromImage(input_image="imgs/rgb.png", output_image_path="imgs/out.png")
    boxl = [0, 0, 0, 0]
    for eachObject in detections:
        if eachObject["name"] == 'sports ball':
            boxl = eachObject["box_points"]
            box = f'[{boxl[0]} {boxl[1]} {boxl[2]} {boxl[3]}]'
            print(box)
            pb.addUserDebugText(box, [0, 0, 2], [0, 0, 0], lifeTime=3)
            # GLDebugDrawString(xStart,yStart,text);
            # print("x1 y1 - lower left; x2 y2 - upper right")
    return len(detections), boxl
```

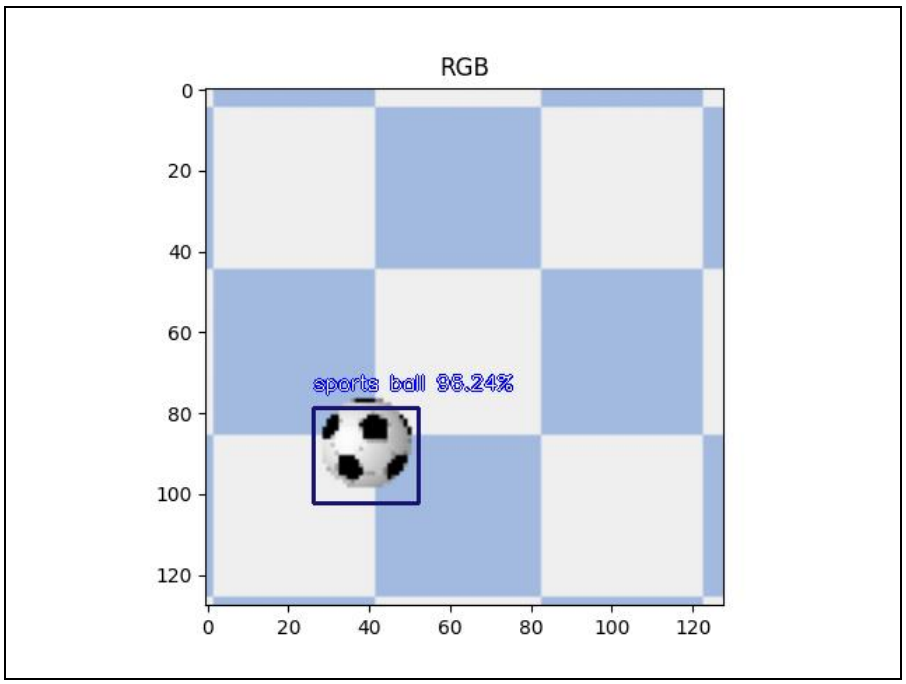

Рисунок 2 – Обработанное изображение

1.2. Перемещение

В симуляции присутствует объект, за которым должно проводиться наблюдение – футбольный мяч с заранее известными формами и габаритами, камера закреплена на дроне и не изменяет своего положения относительно корпуса. Объект постоянно движется в произвольном направлении.

После рендеринга изображения и его обработки камера смещается, основываясь на рассчитанных координатах. Вычисляются координаты центра объекта при его расположении в центре изображения и его текущий центр, с помощью которых вычисляется новое смещение.

Листинг 3.

```
def rotate(box):
    center_ideal_horz = (ideal_coords[2] - ideal_coords[0]) + ideal_coords[0]
    center_ideal_vert = (ideal_coords[3] - ideal_coords[1]) + ideal_coords[1]
    center_horz = (box[2] - box[0]) + box[0]
    center_vert = (box[3] - box[1]) + box[1]

    diff_horz = center_ideal_horz - center_horz
    diff_vert = center_ideal_vert - center_vert

    return diff_horz, diff_vert
```

1.3. Расстояние до объекта

Чтобы определить расстояние от камеры до известного объекта, будем использовать подобие треугольников.

Подобие треугольников выглядит примерно так: допустим, имеется объект с известной шириной `W`. Этот объект помещается на некотором расстоянии `D` от нашей камеры. Далее делается снимок нашего объекта с помощью камеры, и затем измеряется видимая ширина в пикселях `P`. В данном случае используются все те же координаты рамки, полчуенные с помощью `ImageAI`. Это позволяет получить фокусное расстояние `F` нашей камеры:

$$F = (P \times D) / W$$

Далее продолжая перемещать камеру как ближе, так и дальше от объекта, можно применить правило подобия треугольников, чтобы определить расстояние от объекта до камеры, зарание вычислив новую ширину `P`:

$$D' = (W \times F) / P$$

1.4. Симуляция в Pybullet

Стадии слежения за объектом представлены на рис. 3-5.

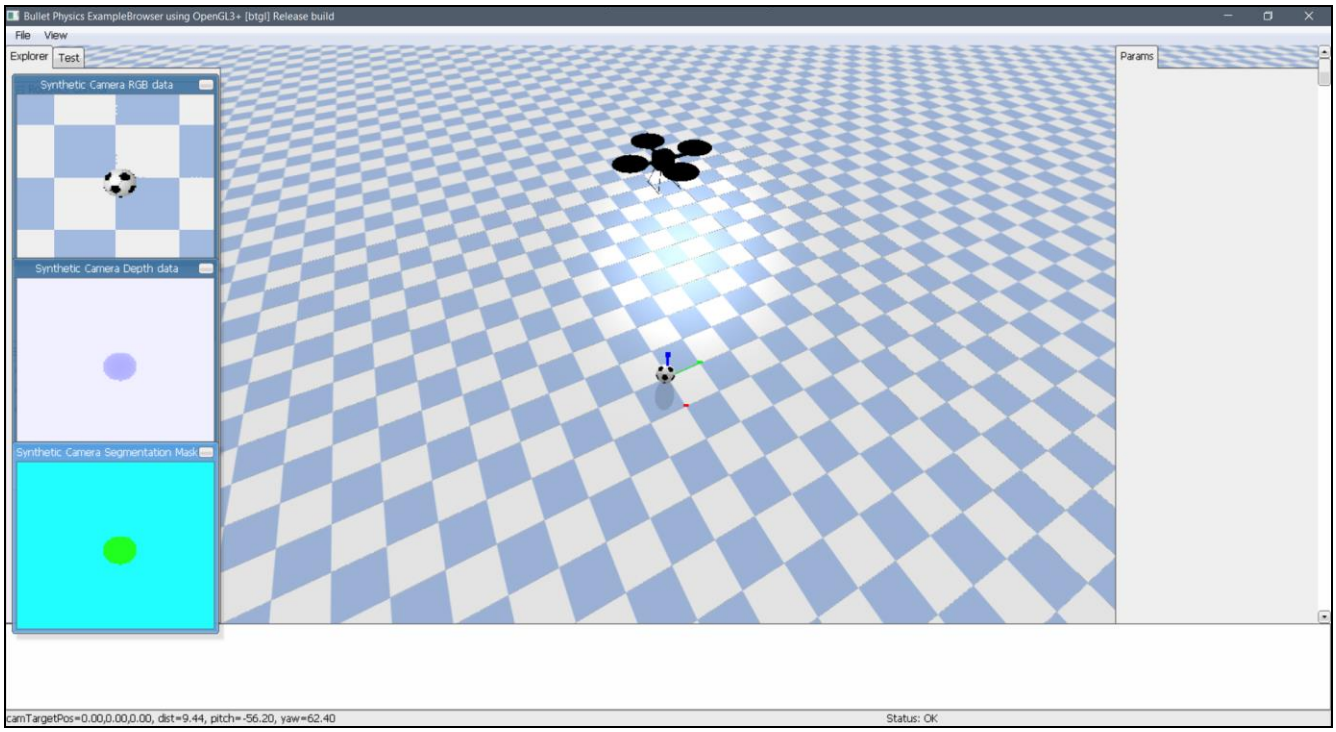

Рисунок 3 – Процесс слежения 1

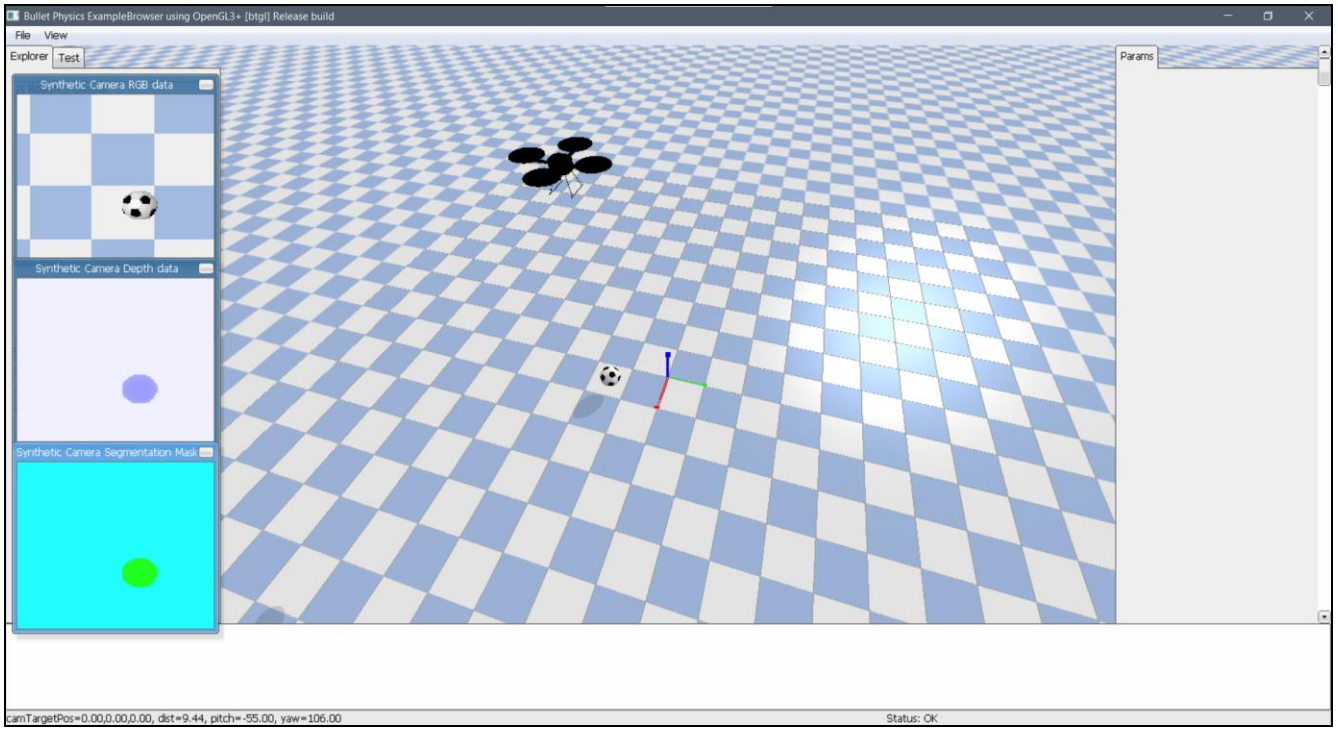

Рисунок 4 – Процесс слежения 2

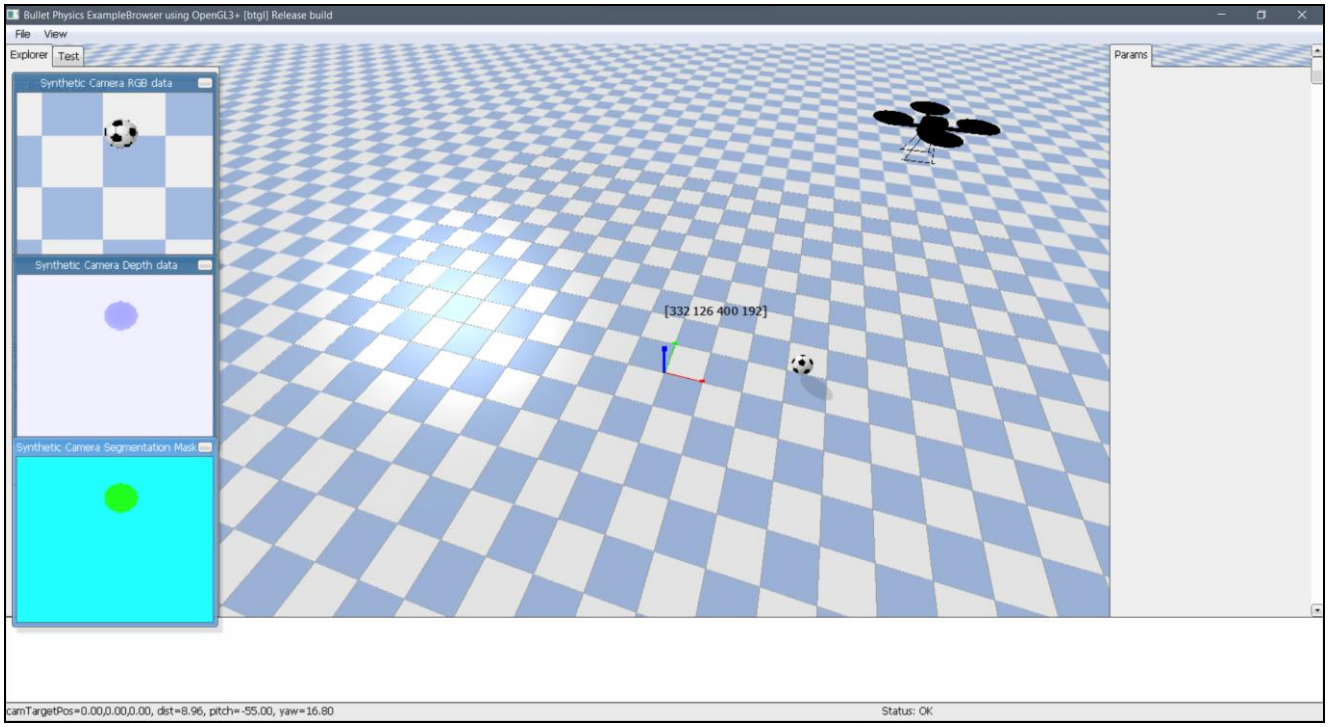

Рисунок 5 – Процесс слежения 3

2. ИСПОЛЬЗОВАННЫЕ ТЕХНОЛОГИИ

- Python
- PyBullet – движок реального времени для физической симуляции
- YoloV3 - алгоритм машинного обучения для распознавания изображений
- ImageAI – библиотека python, которая позволяет легко интегрировать технологии компьютерного зрения в приложение
- Scikit-Learn
- Библиотеки для обработки данных Pandas, NumPy

ОТЗЫВ РУКОВОДИТЕЛЯ

Отзыв руководителя

Студент Киреев Константин Александрович СПбГЭТУ «ЛЭТИ» группы 8383 успешно прошел практику с 30 июня 2021 г. по 20 июля 2021 г. в ИДСТУ СО РАН.

Во время летней практики студент занимался моделированием движения дрона в задаче слежения за движущимся объектом.

Рекомендуемая оценка: отлично.

Руководитель практики: зав. лаб. ИДСТУ СО РАН, к.т.н. Ульянов С.А.

Дата: 20.07.2021 Подпись:

Печать:

Подпись заверяю
Нач. отдела делопроизводства
и организационного обеспечения
ИДСТУ СО РАН
Т.Б. Кононенко
20.07.2021

ЗАКЛЮЧЕНИЕ

В ходе практики были изучены возможности библиотеки PyBullet для решения задачи моделирования. А также были изучены методы анализа и обработки видеоизображений с помощью методов машинного обучения.

СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ

1. Документация PyBullet.
   URL:
   https://github.com/bulletphysics/bullet3/tree/master/docs/pybullet_quickstart_guide
2. Документация ImageAI.
   URL: https://github.com/OlafenwaMoses/ImageAI
3. Max Christl. Vision-Based Autonomous Drone Control using Supervised Learning in Simulation // IMPERIAL COLLEGE LONDON. 2020, arXiv:2009.04298v1
4. Алгоритм оценки расстояния до объекта.
   URL: https://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/