ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В.ЛОМОНОСОВА»

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(ДИПЛОМНАЯ РАБОТА)
специалиста

МОДЕЛИРОВАНИЕ ТРАЕКТОРИИ ДВИЖЕНИЯ КВАДРОКОПТЕРА ПО ВИЗУАЛЬНОЙ ИНФОРМАЦИИ С КАМЕР

Выполнил студент
604 группы
Золотов Глеб Владимирович

_____

подпись студента

Научный руководитель:
научный сотрудник
Шокуров Антон Вячеславович

_____

подпись научного руководителя

Москва

2021

Содержание

1 Введение

В данной работе рассматривается проблема обучения квадрокоптера по изображению с его камеры распознавать прямоугольные рамки и корректировать свой курс с целью пролететь в них.

Тема беспилотного управления летательными аппаратами очень актуальна в наше время: с помощью дронов можно вести сьёмку местности, производить измерения в сельскохозяйственных и географических целях, доставлять предметы, исследовать труднодоступные места. Например, дрон-доставщик может долететь до нужного дома с помощью GPS-навигации, но далее, когда нужно подлететь к нужному окну, подъезду или какому-нибудь маркированному пункту, навигация бессильна, приходится полагаться на средства распознавания, один из которых и будет представлен в данной работе. Также средства навигации не позволяют преодолевать неожиданные препятствия, будь то другие транспортные аппараты, птицы, деревья. В этих задачах методы распознавания оказываются очень эффективными и довольно доступными с точки зрения оборудования (достаточно камеры и микропроцессора, не нужно никаких специализированных датчиков), но в данной работе такая задача решаться не будет, можно считать это задачей на будущее.

2 Постановка задачи

В работе рассматривается дрон (квадрокоптер) с закреплённой на нём камерой. На видео с этой камеры, в каждом кадре присутствует, помимо окружающей среды, не менее одной прямоугольной рамки (необязательно полностью), имеющей шахматную бело-зелёную раскраску.

Задача состоит в том, чтобы на видео с камеры дрона найти ближайшую рамку и скорректировать текущий курс с целью пролететь в неё. Для поиска рамок на видео используется *TensorFlow Object Detection API* [1], инструмент, представляющий из себя набор предобученных на наборе MSCOCO нейронных сетей разных архитектур, сравнение некоторых (самых популярных) из них применительно к данной задаче будет приведено в работе.

Для обучения, проверки результатов и тестирования используется среда симулятора с высокой точностью воспроизведения *AirSim* [2], модификация для соревнования *Game Of Drones* [3], организованного *Microsoft*. Это соревнование проводилось в 2019 году на *NeurIPS* - крупнейшей конференции в области искусственного интеллекта. Суть соревнования

такова: в симуляторе два квадрокоптера одновременно проходят трассу, которая представляет из себя последовательность ворот-рамок бело-зелёной шахматной раскраски с лицевой стороны, бело-красной с тыльной стороны и бело-синей с боковых сторон. Победителем считается та команда, чей дрон первым пройдёт всю трассу, при столкновении квадрокоптеров команда обгоняющего дисквалифицируется. Рамки расставлены таким образом, что каждая следующая рамка видна на кадре с камеры дрона при его перпендикулярном пролёте текущей рамки. Целью соревнования является объединение проблем, связанных с состязательным планированием и восприятием в реальном времени, и поощрение сочетания подходов, основанных на обучении и моделировании. Постановка задачи во многом повторяет условия соревнования, за исключением того, что задача распознавания дрона-соперника и изменения траектории в зависимости от его положения с целью избежать столкновения не рассматривается.

3 Основная часть

Решение задачи состоит из двух пунктов: распознавание на кадре с камеры нужной (самой большой) рамки и численный расчёт корректировки курса. Стоит отметить, что решение общее и не зависит от того, используется ли оно на настоящем дроне или в симуляторе, но для полноты картины нужно описать симулятор, с помощью которого были получены все описываемые результаты.

3.1 Описание симулятора

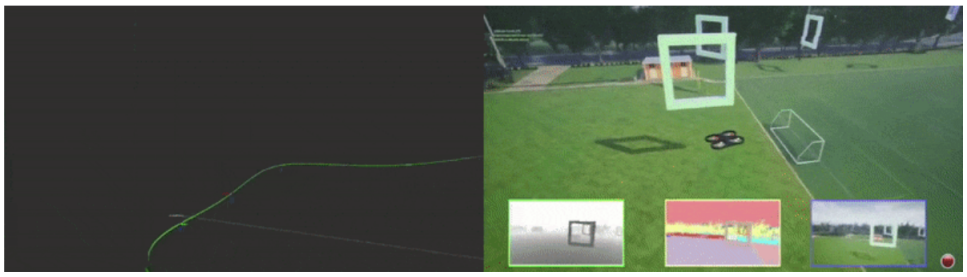
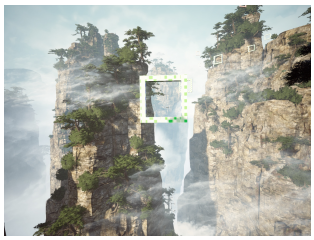


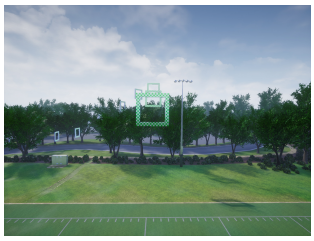
Рис. 1: Симулятор квадрокоптера AirSim

В соревновании *Game Of Drones*, условия коротого были взяты за основу данной работы, предлагается 3 локации в симуляторе *AirSim*:

Soccer Field (представляет из себя футбольное поле с деревьями по периметру), *ZhangJiaJie* (горная местность) и *Building99* (офисное здание). На этих локациях построены трассы трёх уровней сложности: лёгкого (на локации *Soccer Field*, трасса представляет из себя замкнутый круг, рамки расположены на одной высоте относительно земли), среднего (на локациях *Soccer Field* и *ZhangJiaJie*, трасса сложной формы с рамками на разной высоте, но все рамки имеют видимую из предыдущей рамки часть) и сложного (на локации *Building99*, рамки на разных высотах, под разными углами и могут быть полностью не видны, например, могут располагаться за углом коридора).



ZhangJiaJie



Soccer Field

Building99

Рис. 2: Локации симулятора

Обмен информацией с дроном в симуляторе происходит через асинхронное API *airsimneurips* [5], реализованное на *Python*. Ниже приводится список самых используемых и полезных методов:

```
simLoadLevel(level_name="")
```

- загрузка одной из описанных выше трасс

```
enableApiControl(vehicle_name="")
```

- активировать управление дроном с именем *vehicle_ name* по API

```
moveOnSplineAsync(waypoints, vel_max=15.0, acc_max=7.5, add_position_constraint=True,
```

- двигаться по траектории-сплайну, построенному через точки *waypoints*. Данная функция полезна для сбора обучающей выборки, если её использовать вместе со следующей.

```
simListSceneObjects(name_regex='.*')
```

- получить список объектов на локации. При *name_ regex*='Gate.*' получаем список названий всех рамок на трассе. С помощью следующей функции можно получить их координаты.

```
simGetObjectPose(name="")
```

- получить координаты объекта *name* на локации.

```
getMultirotorState(vehicle_name="")
```

- получить информацию о текущих характеристиках дрона с именем *vehicle_ name*. Самое важное из этого - можно получить текущие координаты дрона, его углы наклона, текущие компоненты скорости и ускорения.

```
simGetImages(requests, vehicle_name="")
```

- получить изображение с камеры дрона.

```
moveByRollPitchYawZAsync(roll, pitch, yaw, z, duration, vehicle_name="")
```

- функция, используемая для корректировки курса дрона. Задаются углы крена, тангажа и рыскания, желаемая координата z квадрокоптера и время, которое дрон будет держать эти параметры.

3.2 Распознавание рамки

В данной работе для распознавания рамок будет использоваться нейросетевой подход. Для сбора данных для обучения моделей можно поступить следующим образом: в симуляторе есть возможность перед стартом получить координаты всех рамок, по этим данным построить траекторию-сплайн для движения дрона и сохранять изображения с его камеры во время движения. Таким образом была получена обучающая выборка размером 350 изображений. Для разметки (выделения на изображениях рамок) был использован графический инструмент *LabelImg* [4], сгенерированные файлы были преобразованы из формата *.xml* в *.csv*, далее полученные файлы и исходные изображения были преобразованы в требуемый фреймворком *TensorFlow* формат *.tfrecord*. Стоит отметить, что обучающую выборку можно многократно увеличить, используя технологию морфинга, но данной выборки было достаточно для удовлетворительного качества распознавания с помощью одной из моделей, поэтому увеличение обучающей выборки не производилось.

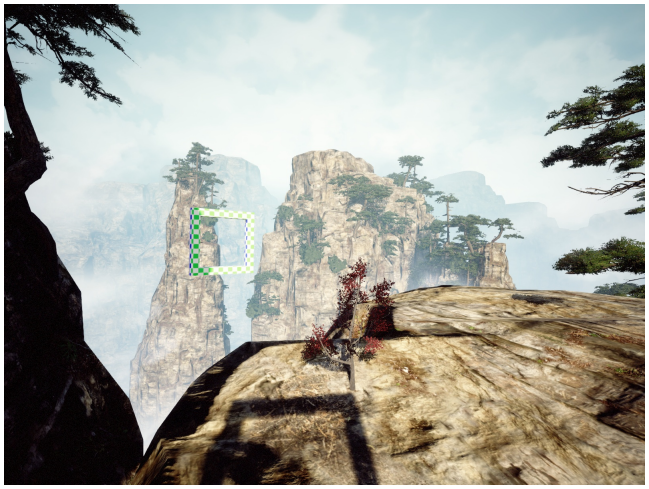


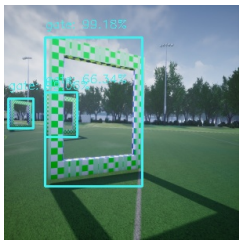
Рис. 3: Пример исходного изображения

Все модели обучались на одном и том же наборе данных, описанном выше, обучающая выборка состояла из 315 изображений, тестовая из 35 изображений. Время для обучения также совпадает и составляет 10 часов. Обучение и тестирование проводилось на одном и том же компьютере. Результаты для простой трассы приведены в таблице:

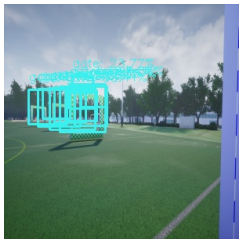
Название модели	Качество, %	Время на один кадр, с
SSD MobileNet v1	93,26	0,35
SSD MobileNet v2	<80	0,35
Faster RCNN Inception v2	89,68	1,35
Faster RCNN ResNet50	<80	5,05

Полученные результаты говорят о том, что для сетей *SSD MobileNet v2* и *Faster RCNN ResNet50* обучающая выборка слишком мала, они не дают необходимого для работы уровня точности, а *Faster RCNN Inception v2* и *Faster RCNN ResNet50* из-за своей архитектуры работают достаточно долго на данной машине, поэтому плохо подходят для потоковой обработки видеоинформации. У *Faster RCNN Inception v2* достаточное качество, но много ложных срабатываний (на Рис. 4). Поэтому в данном случае лучшим выбором будет модель *SSD MobileNet v1*. Из-за свойств данной модели даже для малого порога достоверности (в данной работе использовался порог достоверности в 20%) ложных срабатываний пренебрежительно мало, поэтому дополнительная обработка результатов распознавания не требуется.

Замечу, что была бы очень полезна обработка другого рода. Результатом работы сети является обрамляющий рамку на изображении прямоугольник. Очевидно, что прямоугольная рамка в кадре далеко не всегда имеет форму прямоугольника, поэтому заметным улучшением распознавания был бы механизм поиска углов рамки внутри распознанной сетью прямоугольной области. Данный механизм можно реализовать двумя способами: используя традиционные методы компьютерного зрения, такие как детектирование границ, размытие и пороговые преобразования по цвету, или ещё раз воспользоваться предобученной нейронной сетью, чтобы дообучить её на изображениях с размеченными углами. Данный шаг не был выполнен по причине того, что получаемых результатов достаточно для рассмотренного далее механизма корректировки курса на трассах лёгкого и, частично, среднего уровней сложности.



SSD MobileNet v1



Faster RCNN Inception v2

Рис. 4: Результаты распознавания

3.3 Расчёт корректировки курса

Квадрокоптер управляется с помощью трёх углов: *крена (roll)*, *тангажа (pitch)*, и *рыскания (yaw)*. Каждый из них отвечает за вращение вокруг одной из осей координат. Движение по оси Ox происходит за счёт изменения угла тангажа, по Oy за счёт изменения угла крена, угол рыскания отвечает за повороты дрона вокруг перпендикулярной дрону оси, то есть за вращение. Таким образом, поворот дрона в горизонтальной плоскости возможен двумя способами: при одновременно ненулевых углах тангажа и крена или при изменении угла рыскания. Так как в данной задаче на дроне закреплена камера, то необходимо, чтобы она всегда была сонаправлена с движением дрона, поэтому при поворотах в горизонтальной плоскости будут использоваться оба способа одновременно.

Получив прямоугольник, обрамляющий ближайшую рамку (самый большой по площади), можно рассчитать его центр. Изменение курса

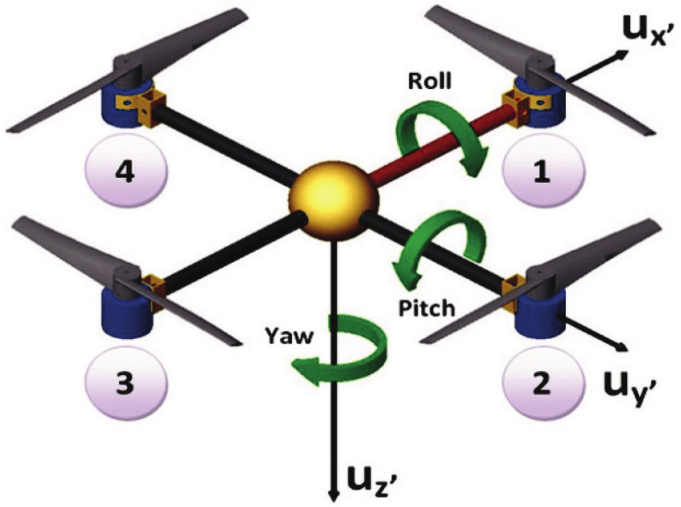


Рис. 5: Углы управления квадрокоптером

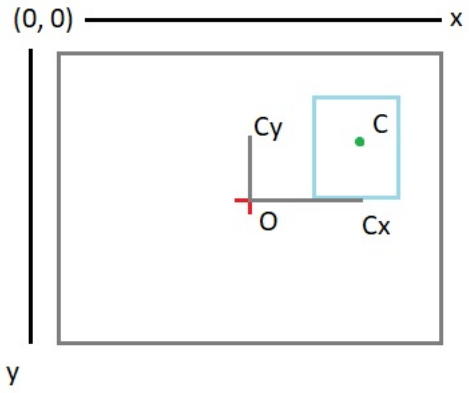


Рис. 6: Центр рамки в точке C

квадрокоптера представляет собой поворот в горизонтальной плоскости и изменение высоты полёта.

За изменение высоты полёта отвечает координата z, изменение пропорционально длине отрезка OCy на Рис. 6.

Поворот в горизонтальной плоскости - комбинация изменения угла крена и рыскания, оба изменения пропорциональны длине отрезка OCx на Рис. 6, а так же отношению максимальной из сторон рамки к размеру изображения (эта величина показывает, насколько близко дрон находит-

ся к рамке, чем ближе - тем больше должны измениться углы в данный момент) и скорости дрона (аналогично с расстоянием, чем больше скорость, тем больше надо изменить углы).

Угол тангажа отвечает за ускорение дрона. Вместе с тем, для каждой трассы существует скорость, при которой квадрокоптер пролетает расстояние между соседними рамками, не успев скорректировать курс, то есть повернуть. Обозначим эту величину как v_{max}, при её достижении необходимо закончить ускорение, дальнейшее увеличение скорости приведёт к нежелательной ситуации невозможности пролёта в цели.

Таким образом, получаем формулы:

$$roll = \vec{OCx} * K * ln((v+1)e) * k_1$$

$$pitch = \begin{cases} k_2 & v < v_{max} \\ 0 & v >= v_{max} \end{cases}$$

$$\Delta yaw = \vec{OCx} * K * ln((v+1)e) * k_3$$

$$\Delta z = \vec{OCy} * k_4$$

, где $K = max(x/w, y/h), k_1, k_2, k_3, k_4$ - подобранные опытным путём коэффициенты.

Опробуем предложенную модель на трассе лёгкого уровня на локации *Soccer Field* в сравнении с базовым алгоритмом полёта дрона, при котором ему на старте известны координаты рамок, и он движется по траектории-сплайну, проходящей через все рамки, с ограничением скорости. Трасса лёгкого уровня была выбрана для сравнения по той причине, что на трассах среднего и особенно сложного уровня в некоторых случаях оказывается существенным действительное расположение рамки относительно дрона, информации об обрамляющем прямоугольнике недостаточно, как было отмечено ранее в пункте о распознавании рамки на изображении. Результаты запусков и полученные траектории ниже:

	Время прохождения трассы, с
Базовый алгоритм	54,44
Предложенная модель	51,98

Можно заметить (на Рис. 7), что траектория получается неровной, присутствуют повороты в обе стороны. Этот эффект происходит из-за задержки между кадрами: по первому кадру курс был скорректирован,

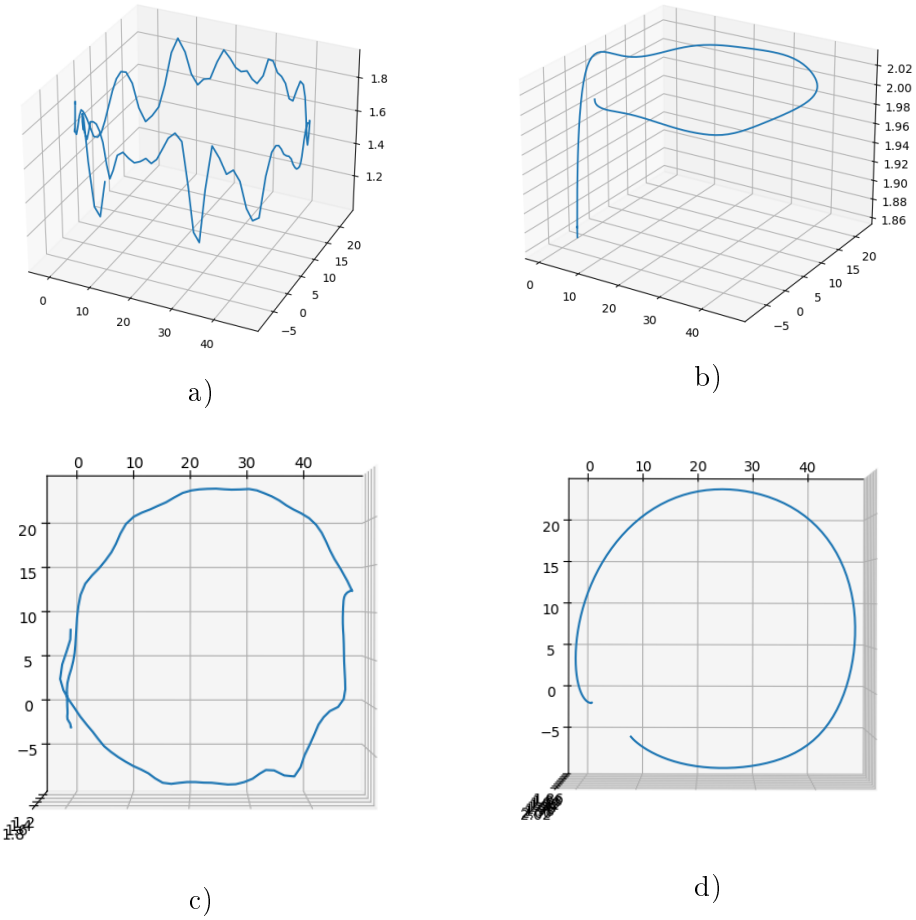


Рис. 7: Полученные траектории движения дрона: a) Предложенная модель, b) Базовый алгоритм, c) Предложенная модель, плоскость XY, d) Базовый алгоритм, плоскость XY.

но к тому моменту, как будет обработан второй кадр, иногда дрон продолжает поворот дольше, чем это требуется, поэтому по второму кадру делает обратную корректировку.

4 Выводы

В предложенном решении не рассматривается положение рамки относительно дрона. Это может стать проблемой, например, если квадрокоптер

подлетает к рамке сбоку, так и происходит на трассах средней и высокой сложности. Можно считать это задачей на будущее, данное улучшение способно серьёзно улучшить получаемые результаты.

Зелёный цвет прямоугольников рамки - условность, можно заменить на любой другой. Это позволяет, с помощью сравнительно небольших модификаций, научить дрон залетать в рамку с другой стороны, например, как в используемом симуляторе *AirSim* [2], где лицевая сторона рамки имеет зелёный цвет, задняя - красный, а боковая - синий. Если надо пролететь в рамку с другой стороны, можно по одному кадру вычислить траекторию, и следить только за появлением препятствий, без распознавания рамки, до тех пор, пока она не станет видна с нужной стороны.

Решение не привязано ко внешнему виду рамки, поэтому цели могут выглядеть как угодно, главное - подготовить обучающую выборку нужных объектов, разметить её и обучить модель. По этой причине результат этой работы может быть перенесён в настоящие, повседневные задачи, скажем, в задачу пролёта дрона в окно.

Дальнейшее развитие этой работы я вижу в возможности распознавать и преодолевать препятствия на пути к цели, будь то деревья, ветки, стены коридоров, холмы и т.д. Это сделает работу более полезной, с большим количеством потенциальных применений.

Список литературы

[1] https://www.tensorflow.org/

[2] https://microsoft.github.io/AirSim/

[3] https://microsoft.github.io/AirSim-NeurIPS2019-Drone-Racing/

[4] https://github.com/tzutalin/labelImg

[5] https://pypi.org/project/airsimneurips/