



Алексей
Зотов



Дмитрий
Пудалов



Тимофей
Бацын



Дмитрий
Белокрылов

КОМАНДА

FIRSTTEAM

ПРЕДСТАВЛЯЕТ ИННОВАЦИОННЫЙ ПРОЕКТ:

Автономная система обнаружения и обезвреживания БПЛА

актуальная проблема

поскольку инфраструктура всё чаще подвергаются атакам с применением БПЛА, мы решили остановиться на проблеме, связанной с защитой важных объектов

ВАРИАНТЫ РЕШЕНИЯ

Заглушение связи БПЛА с оператором по средствам радио-электронной борьбы

Обнаружение посредством КЗ с последующим противодействием

Запуск само-управляемого дрона, ориентирующегося на источник звука

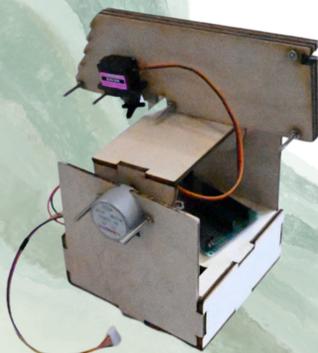
РОБОТИЗИРОВАННЫЕ РЕШЕНИЯ

1 вариант

Мы использовали в качестве условного дрона деталь определённого цвета, которого нет на фоне. Камера была статичной, вращался только исполнительный блок. Метод обнаружения: поиск скопления пикселей определённого цвета и наведение на геометрический центр скопления

Минусы:

- сильно зависит от освещения;
- малый угол обзора;
- работает не на любом фоне;
- не всегда правильно наводится;
- ненадёжная конструкция

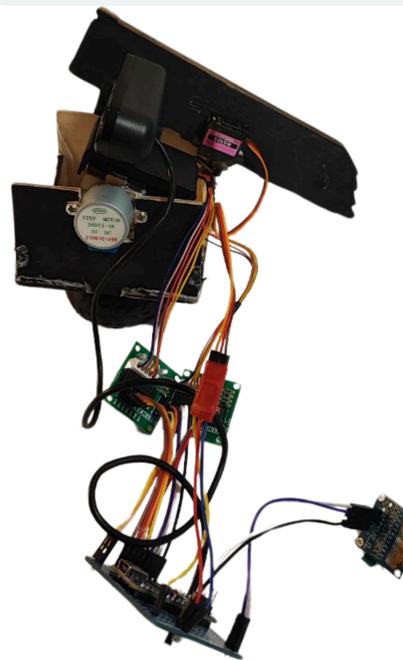


2 вариант

Мы усилили корпус, развели плату на макетной основе, увеличили боезапас относительно первой версии. Установили камеру на исполнительный блок, что сильно увеличило угол обнаружения и точность наведения

Минусы:

- всё ещё сильно зависит от освещения;
- работает не на любом фоне;
- не всегда правильно наводится;
- работает только на условной цели;
- атака происходит вне зависимости от расстояния до условного БПЛА



ОБУЧЕНИЕ НЕЙРОСЕТИ

1 этап

- фотографируем дрон с разных ракурсов, в разных помещениях с разным освещением;
- размечаем на классы (drone/air);
- создаём .txt-файлы с коорд. выделенных областей объектов

2 этап

- делим на пакеты train (обучение) и test (тестирование);
- подключение docker-контейнера для развертывания машинного обучения;
- запуск машинного обучения

3 этап

- проверка результатов обучения на пакете test;
- дополнительное обучение нейросети;
- интеграция результатов обучения в основной python-код



консультанты

проекта:

Грачёв Алексей Анатольевич - arduino;
Глушков Дмитрий Вадимович - нейросети;
Гринберг Евгений Евгеньевич - нейросети



финальный вариант

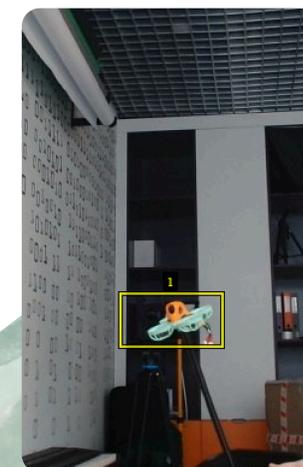
- обнаруживает реальный дрон, а не условный объект;
- использует машинное обучение;
- способен отличить БПЛА от иных объектов;
- может поддерживать разные классы объектов, находящихся в зоне видимости;
- не зависит от освещения;
- работает на любом фоне;
- гораздо лучше наводится;
- атака зависит от расстояния до БПЛА

ПРОГРАММНАЯ ЧАСТЬ

- от нейросети мы получаем координаты дрона;
- в python данные координаты преобразуются в количество шагов, необходимое мотору для наведения по осям;
- отсылаем команды наведения на arduino;
- робот следит за дроном с использованием ПИД-регулятора
- при определенном расстоянии до дрона робот производит залп

АППАРАТНАЯ ЧАСТЬ

- Arduino Nano управляет исполнительным механизмом;
- 2 шаговых мотора отвечают за наведение;
- сервомотор обеспечивает стрельбу
- датчик расстояния определяет расстояние до предполагаемой цели;
- веб-камера передаёт изображение в ЦВС;
- центральная вычислительная система в виде ноутбука



```
def img=False):
weights, view_img, save_txt, imgs, trace = opt.s
not opt.nosave and not source.endswith('.txt')
source.isnumeric() or source.endswith('.txt') on
sp://', 'rtsp://', 'http://', 'https://'))

ories
Path(increment_path(Path(opt.project) / opt.na
r / 'labels' if save_txt else save_dir).mkdir(par

ize
img()
select_device(opt.device)
device.type != 'cpu' # half precision only support

odel
attempt_load(weights, map_location=device) # loa
int(model.stride.max()) # model stride
check_img_size(imgsz, s=stride) # check img_size

l = TracedModel(model, device, opt_img_size)

l.half() # to FP16

stage_classifier
= False
ify:
lc = load_classifier(name='resnet101', n=2) # in
```

cls	total	Labels	img_size
081426	0.02894	4	640: 100%
	P	R	mAP@.5 mAP@.5: .95: 100%
	0.71	0.864	0.894 0.625
081529	0.02859	8	640: 100%
	P	R	mAP@.5 mAP@.5: .95: 100%
	0.679	0.858	0.837 0.616
081543	0.02979	9	640: 100%
	P	R	mAP@.5 mAP@.5: .95: 100%
	0.7	0.859	0.851 0.623
080138	0.02838	0	640: 100%
	P	R	mAP@.5 mAP@.5: .95: 100%
	0.684	0.858	0.829 0.616
081176	0.02834	0	640: 100%
	P	R	mAP@.5 mAP@.5: .95: 100%
	0.783	0.897	0.882 0.626
080197	0.01983	4	640: 100%
	P	R	mAP@.5 mAP@.5: .95: 100%
	0.695	0.842	0.838 0.625
08021	0.02817	1	640: 100%
	P	R	mAP@.5 mAP@.5: .95: 100%
	0.682	0.867	0.836 0.622
081169	0.02827	29	640: 71%