

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

ОТЧЕТ

по производственной практике

**Тема: Разработка алгоритма автоматической посадки БПЛА с
применением компьютерного зрения без использования GPS**

Студент гр. 2303

Мышкин Н.В.

Руководитель (от СПбГЭТУ
“ЛЭТИ”)

Заславский М.М.

Руководитель (от АО “ОКБ
Электроавтоматика”)

Костишин М.О.

Санкт-Петербург
2025

ЗАДАНИЕ
НА ПРОИЗВОДСТВЕННУЮ ПРАКТИКУ

Студент Мышкин Н.В.

Группа 2303

Тема практики: Разработка алгоритма автоматической посадки БПЛА с применением компьютерного зрения без использования GPS

Задание на практику:

Разработать и реализовать алгоритм для автоматического распознавания посадочной площадки БПЛА на основе обработки видеоизображений с бортовой камеры. Алгоритм должен включать следующие этапы:

- 1) Захват и предварительная обработка кадров.
- 2) Сегментация изображения.
- 3) Идентификация площадки по форме.
- 4) Расчет координат центра и угла ориентации площадки.

Сроки прохождения практики: 10.02.25 – 06.06.25

Дата сдачи отчета: 19.05.25

Дата защиты отчета: 19.05.25

Студент		Мышкин Н.В.
Руководитель (от СПбГЭТУ “ЛЭТИ”)		Заславский М.М.
Руководитель (от АО “ОКБ Электроавтоматика”)		Костишин М.О.

АННОТАЦИЯ

Целью производственной практики является разработка алгоритма автоматической посадки БПЛА с использованием компьютерного зрения без GPS. Основные задачи включали обработку видеоизображений, распознавание посадочной площадки. Для реализации алгоритма применены библиотеки OpenCV и NumPy, а также методы фильтрации и сегментации изображений. Проведено тестирование для разных посадочных площадок (круг, квадрат, прямоугольник, что подтвердило работоспособность решения.

Практика позволила закрепить навыки работы с большими проектами и решением реальных задач в области авионики. Результаты могут быть использованы для дальнейшего развития автономных систем посадки БПЛА.

SUMMARY

The aim of the internship was to develop an algorithm for automatic UAV landing using computer vision without GPS. The main tasks were video image processing and recognizing the landing site. OpenCV and NumPy libraries, as well as methods of image filtering and segmentation are used to implement the algorithm. Testing was carried out for different landing sites (circle, square, rectangle), which confirmed the performance of the solution.

Practice allowed to consolidate the skills of working with large projects and solving real problems in the field of avionics. The obtained results can be used for further development of UAV autonomous landing systems.

СОДЕРЖАНИЕ

Введение	5
1. Выбранные библиотеки для реализации алгоритма	6
1.1. Библиотека OpenCV 4.11.0	6
1.2. Библиотека NumPy 1.26.4	6
1.3. Библиотека Matplotlib 3.9.2	7
1.4. Библиотека Imutils 0.5.4	7
1.5. Библиотека Os 3.11.10	8
1.6. Библиотека Datetime 3.11.10	8
2. Описание алгоритма	9
2.1. Класс LandingZoneDetector	9
2.2. Метод __init__(min_contour_area, target_aspect_ratio, tolerance)	9
2.3. Метод preprocess_frame(frame)	9
2.4. Метод detect_edges(image)	10
2.5. Метод find_contours(edged_image)	10
2.6. Метод filter_contours(contours)	10
2.7. Метод calculate_contour_center_and_orientation(contour)	11
2.8. Метод _debug_visualization(original, processed, edged, contour, center, angle, frame_count)	11
2.9. Метод detect_landing_zone(frame, debug, frame_count)	11
2.10. Функция main()	12
3. Сборка и тестирование алгоритма	13
3.1. Сборка	13
3.2. Тестирование	15
Заключение	19
Список использованных источников	20
Приложение А. Название приложения	21

ВВЕДЕНИЕ

Основная цель производственной практики заключается в приобретение практического опыта в разработке программного обеспечения для авиационных систем, изучение современных технологий обработки данных и компьютерного зрения, а также интеграция решений в реальные проекты предприятия.

Основной задачей является разработка алгоритма автоматической посадки БПЛА с использованием компьютерного зрения без GPS, включая обработку видеоизображений, распознавание посадочной площадки, с последующей адаптацией решения под требования предприятия. А также тестирование алгоритма на различных входных данных.

Реализация основной задачи началась с поиска решения какое может быть применено к данной задаче. Изначально было решено ознакомиться со всеми теоретическими материалами предоставленными организацией. Позже было принято решение использовать библиотеку OpenCV, так как в ней оказалось весь необходимый функционал для написания алгоритма.

Также в ходе выполнения поставленной задачи было принято решение сделать расширенные возможности для отслеживания ошибок, так как при первых тестовых видео не всегда получается получить желаемый результат.

1. ВЫБРАННЫЕ БИБЛИОТЕКИ ДЛЯ РЕАЛИЗАЦИИ АЛГОРИТМА

1.1. Библиотека OpenCV 4.11.0

OpenCV (Open Source Computer Vision Library) — это библиотека для компьютерного зрения, обработки изображений и машинного обучения. Она поддерживает множество алгоритмов и предоставляет инструменты для работы с видео и изображениями. В данном коде OpenCV используется для следующих задач:

- 1) Работа с изображениями и видео функции (`cv2.VideoCapture()` – чтение видео из файла или камеры, `cv2.VideoWriter()` – запись видео в файл, `cap.read()` – получение очередного кадра из видео, `cv2.imshow()` – отображение изображения в окне, `cv2.waitKey()` – ожидание нажатия клавиши (используется для управления воспроизведением).
- 2) Обработка изображений (`cv2.cvtColor()` – преобразование цветового пространства, `cv2.GaussianBlur()` – размытие изображения для уменьшения шума, `cv2.Canny()` – детекция границ с помощью алгоритма Кэнни).
- 3) Поиск контуров (`cv2.findContours()` – поиск контуров на бинарном изображении, `cv2.contourArea()` – вычисление площади контура, `cv2.contourArea()` – вычисление площади контура и др.).
- 4) Анализ геометрии контуров (`cv2.moments()` – вычисление моментов контура (используется для нахождения центра масс), `cv2.drawContours()` – отрисовка контуров на изображении, `cv2.circle()` – рисование круга (используется для отметки центра), `cv2.line()` – рисование линии (используется для отображения ориентации), `cv2.putText()` – добавление текста на изображение.

1.2. Библиотека NumPy 1.26.4

NumPy (Numerical Python) — это фундаментальная библиотека для научных вычислений в Python. Она предоставляет мощные инструменты для

работы с многомерными массивами, математическими функциями и линейной алгеброй. В данном коде NumPy используется для следующих задач:

- 1) Работа с массивами. NumPy основан на объекте ndarray (N-dimensional array), который позволяет эффективно хранить и обрабатывать большие объемы данных.
- 2) Математические вычисления. np.arctan2(y, x) — вычисляет арктангенс от y/x с учетом квадранта (используется для определения угла ориентации контура), np.degrees() – преобразует радианы в градусы, np.median() - используется для вычисления медианы (среднего значения в отсортированном наборе данных).

1.3. Библиотека Matplotlib 3.9.2

Matplotlib – это библиотека для создания статических, анимированных и интерактивных визуализаций в Python. В данном коде Matplotlib используется для следующих задач:

- 1) Создание составных изображений (plt.figure() – создает новую фигуру заданного размера, plt.subplot() – Добавляет подграфик (subplot) в сетку).
- 2) Отображение и сохранение изображений (plt.imshow() – отображает изображение на текущем subplot, plt.title() – добавляет заголовок к subplot, plt.tight_layout() – автоматически отступы между subplots, чтобы избежать наложений, plt.savefig() – сохраняет фигуру в файл).

1.4. Библиотека Imutils 0.5.4

Imutils — это набор удобных функций-утилит для упрощения работы с OpenCV. Она не добавляет новой функциональности, но делает стандартные операции более удобными и читаемыми. В данном коде используется всего одна функция из этой библиотеки, но она решает важную проблему совместимости между разными версиями OpenCV. В данной ситуации Функция cv2.findContours() в разных версиях OpenCV возвращает разные структуры данных grab_contours() – решает эту проблему.

1.5. Библиотека Os 3.11.10

Модуль os предоставляет кроссплатформенные функции для взаимодействия с операционной системой. В данном коде он используется для организации работы с файлами и директориями:

- 1) Создание директорий (os.makedirs())
- 2) Формирование путей к файлам (os.path.join())

1.6. Библиотека Datetime 3.11.10

Модуль datetime используется для генерации уникальных имен файлов с временными метками. В данном коде он используется для организации работы с файлами и директориями:

- 1) Форматирование времени (strftime())

2. ОПИСАНИЕ АЛГОРИТМА

2.1. Класс LandingZoneDetector

Класс LandingZoneDetector предназначен для обнаружения зоны посадки на видео. Он использует различные методы обработки изображений и компьютерного зрения для выполнения этой задачи. Давайте рассмотрим каждый метод класса более подробно.

2.2. Метод __init__(min_contour_area, target_aspect_ratio, tolerance)

Метод `__init__` инициализирует параметры детектора. Он принимает три параметра: `min_contour_area`, `target_aspect_ratio` и `tolerance`. `min_contour_area` определяет минимальную площадь контура, которая по умолчанию составляет 500 пикселей. Это значение помогает отфильтровать мелкие объекты, которые не представляют интереса. `target_aspect_ratio` задает ожидаемое соотношение сторон площадки, например, 1.0 для квадрата. Это значение помогает определить форму объекта, который мы ищем. `tolerance` определяет допустимое отклонение от `target_aspect_ratio`, что позволяет учитывать небольшие искажения формы. Метод сохраняет переданные параметры в атрибуты класса, которые будут использоваться в дальнейших вычислениях и обработках.

2.3. Метод preprocess_frame(frame)

Метод `preprocess_frame` выполняет предварительную обработку кадра. Он принимает входной кадр в формате BGR и возвращает размытое изображение в оттенках серого. Метод конвертирует кадр в оттенки серого с использованием `cv2.COLOR_BGR2GRAY`, что упрощает дальнейшую обработку, так как цветовая информация не всегда необходима для детекции границ. Затем применяется Гауссово размытие с ядром 9×9 для уменьшения шума и улучшения детекции границ. Размытие помогает сгладить мелкие детали и улучшить качество обнаружения границ.

2.4. Метод detect_edges(image)

Метод `detect_edges` выполняет детекцию границ оператором Кэнни. Он принимает изображение в оттенках серого и возвращает бинарное изображение с границами. Метод вычисляет медиану интенсивности пикселей, что позволяет автоматически определить пороги для алгоритма Кэнни. Нижний порог устанавливается как $(1.0 - 0.33) * \text{медиана}$, а верхний порог — как $(1.0 + 0.33) * \text{медиана}$. Эти пороги помогают определить, какие границы считать значимыми. Затем применяется алгоритм Кэнни, который выделяет границы на изображении.

2.5. Метод findContours(edged_image)

Метод `findContours` выполняет поиск контуров на бинарном изображении. Он принимает изображение с границами и возвращает список контуров. Метод использует `cv2.findContours` в режиме `RETR_EXTERNAL` (только внешние контуры), что позволяет найти только внешние границы объектов. Затем упрощает контуры с `CHAIN_APPROX_SIMPLE`, что уменьшает количество точек в контуре, сохраняя его форму. Стандартизирует вывод с помощью `imutils.grabContours`, что обеспечивает совместимость с различными версиями OpenCV.

2.6. Метод filterContours(contours)

Метод `filterContours` выполняет фильтрацию контуров по площади и форме. Он принимает список контуров и возвращает отфильтрованный список контуров. Метод отбрасывает контуры с площадью меньше `minContourArea` (используя `cv2.contourArea`), что позволяет отфильтровать мелкие объекты. Затем аппроксимирует контуры многоугольниками (используя `cv2.approxPolyDP`), что упрощает форму контура. Проверяет соотношение сторон (`width/height`) с допуском `tolerance`, что позволяет отфильтровать контуры, которые не соответствуют ожидаемой форме.

2.7. Метод calculate_contour_center_and_orientation(contour)

Метод calculate_contour_center_and_orientation выполняет расчёт центра масс и угла ориентации контура. Он принимает контур и возвращает кортеж ((cx, cy), angle). Метод вычисляет моменты контура (используя cv2.moments), что позволяет определить различные характеристики контура, такие как площадь, центр масс и ориентация. Находит центр масс ($cx = m10 / m00$, $cy = m01 / m00$), что позволяет определить центральную точку контура. Рассчитывает угол ориентации ($angle = 0.5 * \arctan2(2 * mu11, (mu20 - mu02))$) в градусах), что позволяет определить угол наклона контура.

2.8. Метод _debug_visualization(original, processed, edged, contour, center, angle, frame_count)

Метод _debug_visualization выполняет визуализацию промежуточных результатов. Он принимает кадры на разных этапах обработки, найденный контур, координаты центра, угол ориентации и номер кадра. Метод создает папку debug_output (используя os.makedirs), что позволяет сохранить результаты визуализации. Рисует контур, центр и ось ориентации на кадре, что позволяет визуально оценить результаты детекции. Сохраняет 4 subplots с результатами (используя plt.savefig), что позволяет сохранить промежуточные результаты для дальнейшего анализа.

2.9. Метод detect_landing_zone(frame, debug, frame_count)

Метод detect_landing_zone является основным методом детекции. Он принимает входной кадр, флаг отладки и номер кадра. Метод возвращает кортеж ((cx, cy), angle, True) при успехе и ((0, 0), 0, False) при ошибке. Метод последовательно вызывает preprocess_frame, detect_edges, find_contours и filter_contours, что позволяет выполнить полный цикл обработки кадра. Для наибольшего контура вычисляет центр и угол, что позволяет определить положение и ориентацию зоны посадки. В режиме debug сохраняет

визуализацию, что позволяет сохранить промежуточные результаты для дальнейшего анализа.

2.10. Функция main()

Функция main() запускает детектор и обрабатывает видео. Она инициализирует LandingZoneDetector, что позволяет настроить параметры детектора. Открывает видео (используя cv2.VideoCapture), что позволяет загрузить видео для обработки. Для каждого кадра вызывает detect_landing_zone, что позволяет выполнить детекцию зоны посадки. Рисует результат (используя cv2.circle и cv2.putText), что позволяет визуально отметить зону посадки на кадре. Сохраняет кадр в видео (используя cv2.VideoWriter), что позволяет сохранить результаты обработки. Закрывает ресурсы (используя cap.release() и out.release()), что позволяет освободить ресурсы после завершения обработки.

3. СБОРКА И ТЕСТИРОВАНИЕ АЛГОРИТМА

3.1. Сборка

Через терминал нужно установить следующие модули, чтобы можно было запустить код:

```
pip install opencv-python numpy matplotlib imutils
```

```
nikolaymyshkin@MacBook-Air-Nikolaj ~ % pip install opencv-python numpy matplotlib imutils
Requirement already satisfied: opencv-python in /opt/homebrew/lib/python3.11/site-packages (4.11.0.86)
Requirement already satisfied: numpy in /opt/homebrew/lib/python3.11/site-packages (1.26.4)
Requirement already satisfied: matplotlib in /opt/homebrew/lib/python3.11/site-packages (3.9.2)
Requirement already satisfied: imutils in /opt/homebrew/lib/python3.11/site-packages (0.5.4)
Requirement already satisfied: contourpy>=1.0.1 in /opt/homebrew/lib/python3.11/site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/homebrew/lib/python3.11/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /opt/homebrew/lib/python3.11/site-packages (from matplotlib) (4.47.2)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/homebrew/lib/python3.11/site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /opt/homebrew/lib/python3.11/site-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=8 in /opt/homebrew/lib/python3.11/site-packages (from matplotlib) (10.2.0)
Requirement already satisfied: pyParsing>=2.3.1 in /opt/homebrew/lib/python3.11/site-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /opt/homebrew/lib/python3.11/site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /opt/homebrew/lib/python3.11/site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)

[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: python3.11 -m pip install --upgrade pip
nikolaymyshkin@MacBook-Air-Nikolai ~ %
```

Рисунок 1 – установка необходимых библиотек

Открываем файл main.py и в строке

`cap = cv2.VideoCapture("test_video_10.mp4")` в качестве аргумента нужно указать название видеофайла, на котором мы хотим прогнать наш алгоритм. Важно что видеофайл должен лежать в той же директории, что main.py и видеофайл должен иметь расширение mp4.

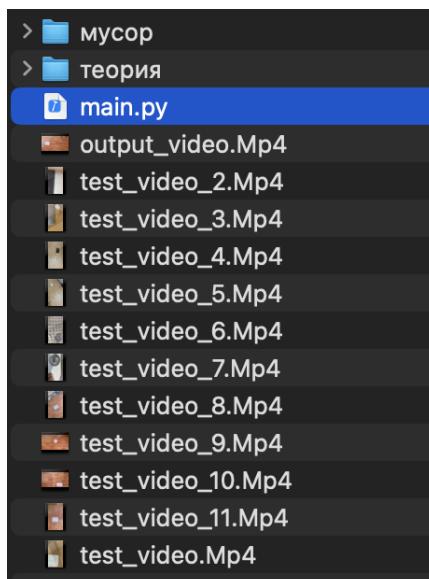


Рисунок 2 – пример как должны располагаться файлы в рабочей директории

Далее если мы хотим запустить алгоритм с отладкой, где у нас будут помимо основного результирующего видео файла еще и промежуточные кадры обработки кадра, нам нужно в строчке `(cx, cy), angle, detected = detector.detect_landing_zone(frame, debug=False, frame_count=0)` сделать `debug=True`. Если мы не хотим запускать алгоритм в режиме отладки, то в строке `(cx, cy), angle, detected = detector.detect_landing_zone(frame, debug=False, frame_count=0)` ничего менять не нужно.

Далее открываем терминал в рабочей директории и запускаем наш алгоритм командой `python3.11 main.py`.

```
nikolaymyshkin@MacBook-Air-Nikolaj PR_PRAKT % python3.11 main.py
2025-05-18 16:25:18.285 Python[64606:5604577] +[IMKClient subclass]: chose IMKClient_Legacy
2025-05-18 16:25:18.285 Python[64606:5604577] +[IMKInputSession subclass]: chose IMKInputSession_Legacy
```

Рисунок 3 – пример запуска алгоритма

После запуска алгоритма сразу же открывается видеофайл, в котором выполняет свою работу алгоритм.

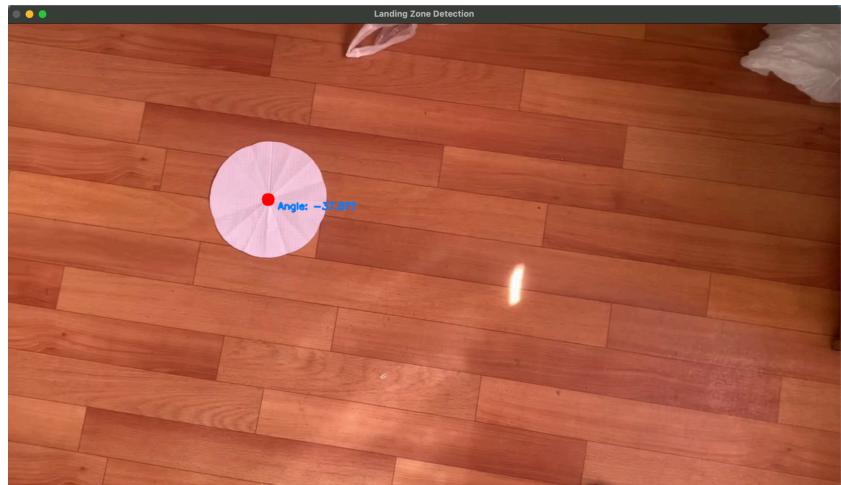


Рисунок 4 – пример работы алгоритма(1 кадр из видеофайла)

После завершения работы алгоритма результирующий видеофайл сохраняется в рабочей директории называется `output_video.Mp4`.

Если мы запустим алгоритм в режиме отладки то помимо видеофайла у нас создастся папка debug_output и туда будет сохраняться каждый кадр из видео.

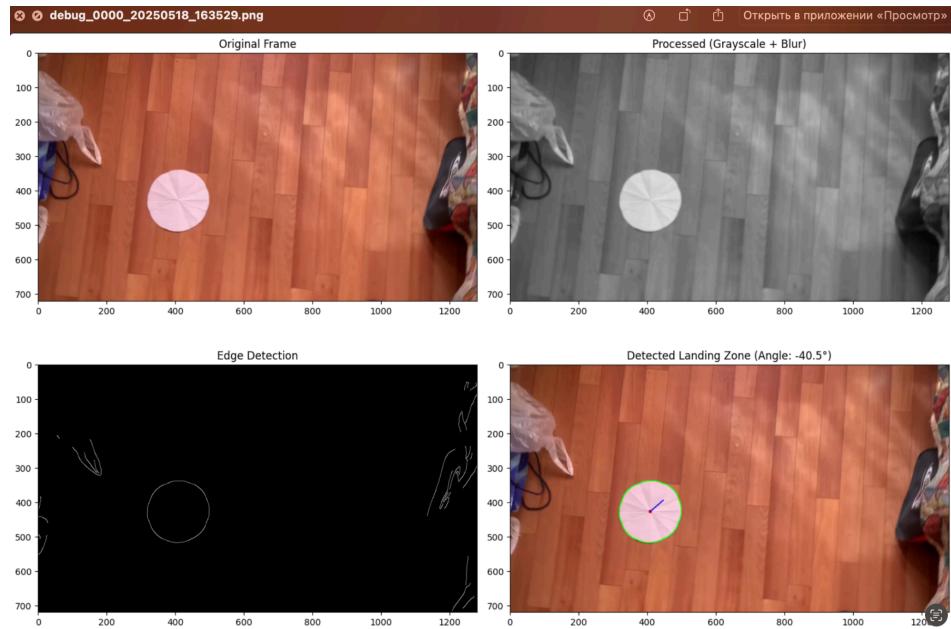


Рисунок 5 – пример работы алгоритма(1 кадр из папки debug_output) содержащий информацию о конкретном кадре

3.2. Тестирование

Для тестирования была написана функция, которая помогает отслеживать каждый кадр, что впоследствии дает понимание, почему в какой-то конкретный момент времени была потеряна точка посадки дрона. Если мы хотим запустить алгоритм с отладкой, где у нас будут помимо основного результирующего видео файла еще и промежуточные кадры обработки кадра, нам нужно в строчке (sx, cy), angle, detected = detector.detect_landing_zone(frame, debug=False, frame_count=0) сделать debug=True. В папке debug_output будет сохраняться каждый кадр из видеофайла обработки, но только на котором была обнаружена посадочная площадка, иначе этот кадр будет проигнорирован. Пример был приведен в разделе 3.1.

Также для тестирования были выбраны освещения и разные фигуры(площадки для посадки). Можно выделить самые удачные из тестов квадрат: test_video_5.mp4, круг: test_video_9.mp4, прямоугольник: test_video_11.mp4

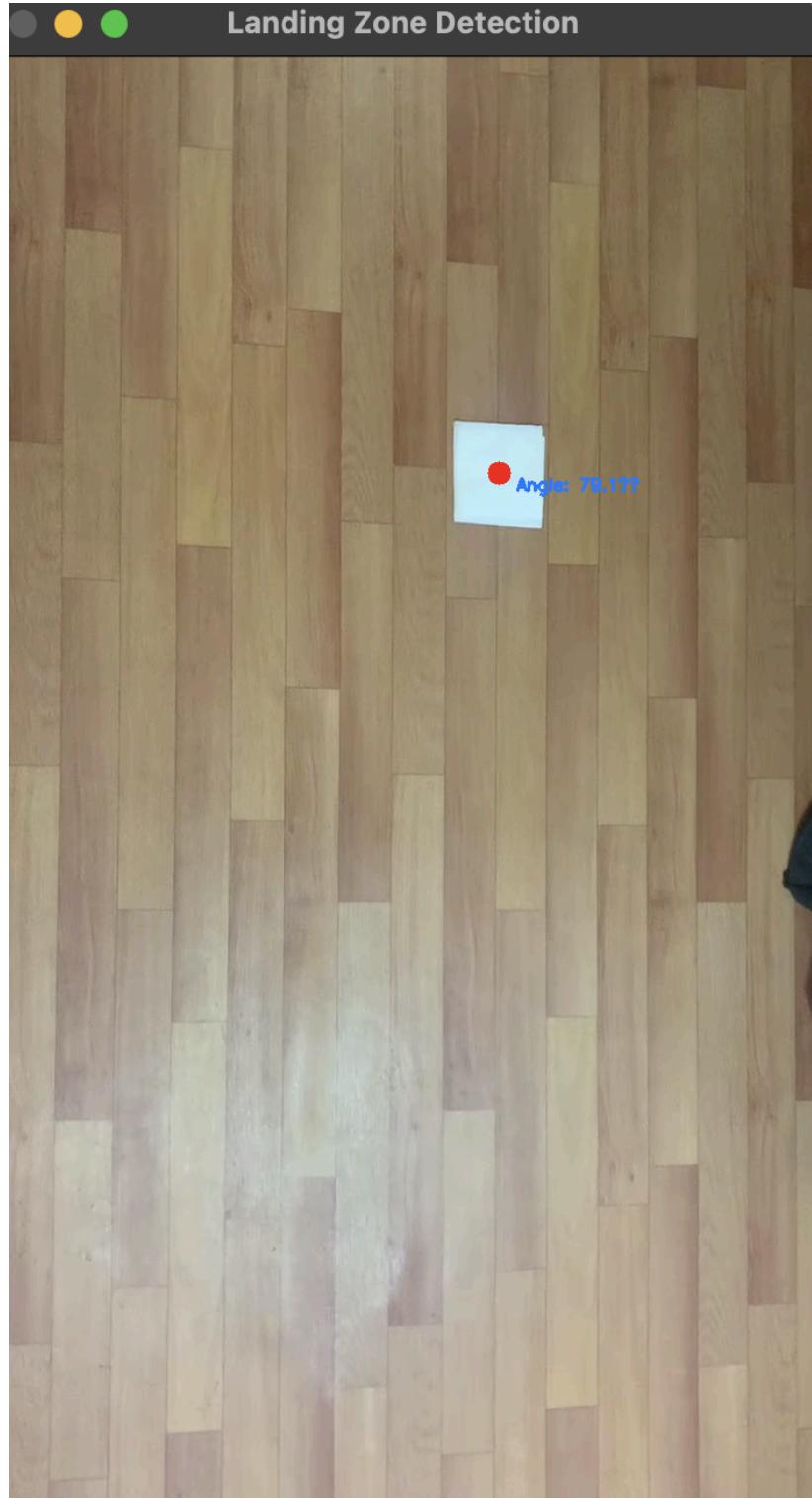


Рисунок 6 – пример работы алгоритма(1 кадр из видеофайла
test_video_5.mp4)

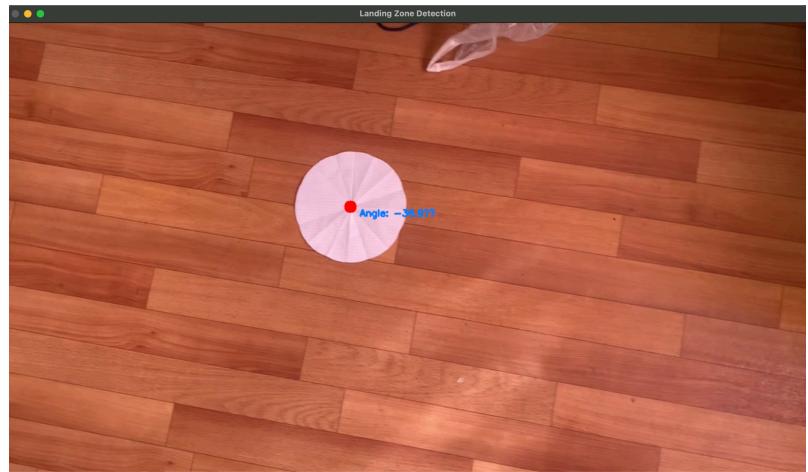


Рисунок 7 – пример работы алгоритма(1 кадр из видеофайла
test_video_9.mp4)



Рисунок 8 – пример работы алгоритма(1 кадр из видеофайла
test_video_11.mp4)

Если запустить алгоритм в режиме отладки, то появится папка debug_output, которая будет содержать png кадры с промежуточными результатами.

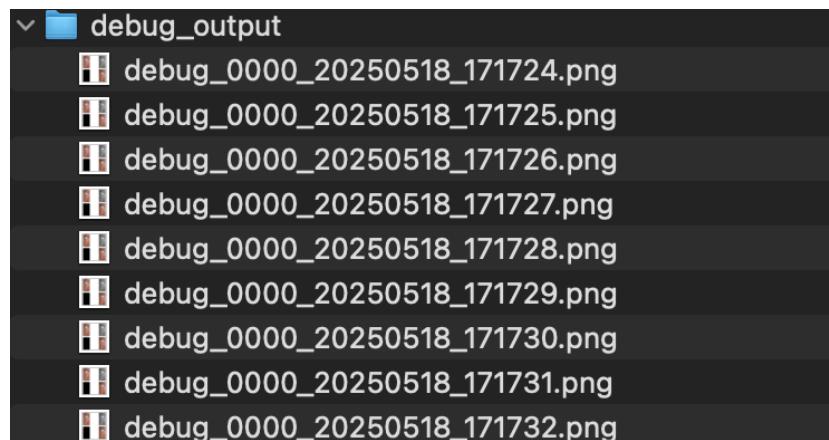


Рисунок 9 – пример того что будет в папке debug_output если запустить алгоритм в режиме отладки

ЗАКЛЮЧЕНИЕ

В ходе производственной практики была успешно разработана и реализована программа для автоматического распознавания посадочной площадки БПЛА с использованием компьютерного зрения. Основные этапы работы включали предварительную обработку кадров, сегментацию изображения, идентификацию площадки по форме и площади, а также расчёт координат центра и угла ориентации. Для достижения поставленной задачи были применены библиотеки OpenCV, NumPy, Matplotlib и другие, что позволило эффективно обрабатывать видеоизображения и анализировать геометрические параметры контуров.

Программа была протестирована на различных типах посадочных площадок (круг, квадрат, прямоугольник) и в разных условиях освещения, что подтвердило ее работоспособность и точность. Режим отладки, включающий сохранение промежуточных результатов, помог детально проанализировать работу алгоритма и выявить возможные ошибки.

Приобретённые навыки работы с большими проектами и современными технологиями обработки изображений будут полезны для дальнейшей профессиональной деятельности в области авионики и компьютерного зрения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Теоретические материалы предоставленные для выполнения производственной практики от компании АО “ОКБ Электроавтоматика” (дата обращения 22.04.25)
2. Документация OpenCV // URL: <https://docs.opencv.org/4.x/index.html> (дата обращения 25.04.25)
3. Документация NumPy // URL: <https://numpy.org/doc/> (дата обращения 25.04.25)
4. Документация Matplotlib // URL: <https://matplotlib.org/stable/index.html> (дата обращения 25.04.25)
5. Документация Imutils // URL: <https://imutils.readthedocs.io/en/latest/> (дата обращения 25.04.25)
6. Документация os // URL: <https://docs.python.org/3/library/os.html> (дата обращения 25.04.25)
7. Документация datetime // URL: <https://pythonworld.ru/moduli/modul-datetime.html> (дата обращения 25.04.25)

ПРИЛОЖЕНИЕ А
НАЗВАНИЕ ПРИЛОЖЕНИЯ