Технические характеристики

Результаты тестирования системы:
• скорость обработки: 20-30 кадров в секунду;
• высота полета: ~ 1 метр;
• скорость облета: ~ 1 м/с;
• продолжительность тестового полета: до 600 секунд (в пределах допустимого заряда батареи).

Преимущества:
• автоматизация процесса выявления дефектов;
• снижение влияния человеческого фактора;
• сокращение времени диагностики участка дороги по сравнению с ручным осмотром;
• возможность масштабирования системы;
• экономия средств за счет автономности.

Методика испытаний:
• тестирование проводилось на имитациях дефектов дорожного покрытия;
• видеопоток анализировался собственной моделью на базе YOLOv8;
• проводились испытания при дневном и искусственном освещении.

Подтверждение результатов:
• видеозапись полета с отмеченными дефектами;
• сопоставление автоматического и ручного анализа.

Приложение 1. Исходный код проекта

```
from ultralytics import YOLO
from djitellopy import Tello
import os
import logging
import cv2
import time

# === Настройки окружения ===
os.environ["OPENCV_FFMPEG_SKIP_FRAME_CHECK"] = "1"

# Отключаем логирование Tello (опционально)
logging.getLogger("djitellopy").setLevel(logging.WARNING)

FRAME_SKIP = 1  # Обрабатывать каждый N-й кадр (ускорение)

# === ИНИЦИАЛИЗАЦИЯ ДРОНА ===
```

```
fly = Tello()
fly.connect()

print(f"Заряд батареи: {fly.get_battery()}%")

# Уменьшить битрейт видео для стабильности при слабом Wi-Fi
fly.set_video_bitrate(1)  # 1 Мбит/с
print("Битрейт видео установлен на 1 Мбит/с")

# Загрузка предобученной модели YOLO
model = YOLO("best_yamki.pt")

# === ВКЛЮЧЕНИЕ ВИДЕОПОТОКА ===
fly.streamon()
print("Видеопоток включён. Ожидание первого кадра...")

frame_read = fly.get_frame_read()

# Ждём первый валидный кадр
while True:
        frame = frame_read.frame
        if frame is not None and frame.size > 0:
        print("Первый кадр получен. Запуск основного цикла.")
      break
        else:
        print("Ожидание кадра...")
      time.sleep(0.1)

# Стабилизация после запуска потока
time.sleep(1)

# === ВЗЛЁТ И ПОДГОТОВКА К ПОЛЁТУ ПО КРУГУ ===
fly.takeoff()
time.sleep(3)
print("Дрон взлетел. Начинаю полёт по кругу...")

# Глобальные переменные
last_annotated_frame = None
frame_counter = 0
start_time_total = time.time()
flight_duration = 40  # Дрон будет лететь по кругу 40 секундq
flying_circle = True

try:
```

```
        while flying_circle:
        loop_start = time.time()

        # Получение кадра
        frame = frame_read.frame
        frame_counter += 1

        if frame is None or frame.size == 0:
        print("Пустой кадр — пропуск...")
         cv2.waitKey(1)
         continue

        my_frame = cv2.resize(frame, (640, 480))

        # Обработка каждого N-го кадра
        if frame_counter % FRAME_SKIP == 0:
        # Модель с трекингом
         results = model.track(
                 my_frame, conf=0.3, persist=True, verbose=False, imgsz=320
        )
        # Модель без трекинга
        # results = model(my_frame, conf=0.3, verbose=False, imgsz=320)
        annotated_frame = results[0].plot()
        last_annotated_frame = annotated_frame
        else:
        last_annotated_frame = my_frame  # fallback

        # Отображение
        display_frame = last_annotated_frame.copy()
        cv2.imshow("Potholes Detection", cv2.cvtColor(display_frame,
cv2.COLOR_BGR2RGB))

        # Управление дроном: полёт по кругу
      # Боковое движение + поворот (вращение)
      fly.send_rc_control(
        left_right_velocity=-25,  # Движение вправо
        forward_backward_velocity=5,
        up_down_velocity=-10,
         yaw_velocity=25,  # Поворот по часовой стрелке
      )

        # Ограничиваем время полёта по кругу
        if time.time() - start_time_total > flight_duration:
        print("Завершение полёта по кругу.")
```

```
            flying_circle = False

        # Выход по 'q'
        if cv2.waitKey(1) & 0xFF == ord("q"):
        flying_circle = False

        # Контроль FPS
        elapsed = time.time() - loop_start
        time.sleep(max(0, 0.033 - elapsed))  # ~30 FPS

except Exception as e:
        print(f"Ошибка: {e}")
finally:
        # === ПОСАДКА И ЗАВЕРШЕНИЕ ===
        fly.send_rc_control(0, 0, 0, 0)  # Остановить движение
        time.sleep(1)
        fly.land()
        time.sleep(3)
        fly.streamoff()
        cv2.destroyAllWindows()
        print(f"Полёт завершён. Заряд батареи: {fly.get_battery()}%")
        fly.end()
```

Приложение 2