Технологии в интеллектуальных системах управления в IoT · 4 курс, бакалавриат

ЛЕКЦИЯ 11

ИСУ на базе МК для смарт-управления сложными производствами и этапы IoT-автоматизации

Пример: система поиска пропавших туристов на Raspberry Pi

Бельгибаев Бауржан Абдрахимович
д.т.н., доцент, кафедра «Искусственный интеллект и big data»
КазНУ им. аль-Фараби

Достоинства предлагаемой системы

Предлагаемая система имеет ряд достоинств по сравнению с классическими методами поиска и спасения пропавших в горах туристов.

Первое, мобильный вариант с квадрокоптером ведет поиск круглосуточно за счет использования камеры ночного видения. Поиск бюджетный по сравнению с вертолетным и пешим вариантами спасения пропавших туристов.

Наличие видеофиксации дисциплинирует туристическую группу и способствует повышению рейтинга турфирмы в лице Заказчиков.

Круглосуточно

Камера ночного видения на квадрокоптере

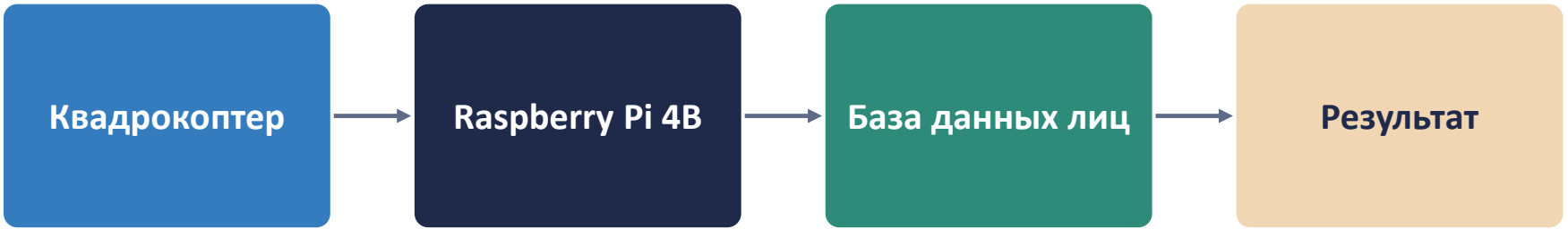
Бюджетно

Дешевле вертолётного и пешего поиска

Видеофиксация

Дисциплина группы и рейтинг турфирмы

Схема 1. Состав ПАК



Локальное хранение биометрии

Данные туристов не уходят в интернет — это снижает юридические риски утечки.

Аппаратная часть: Raspberry Pi 4B

Вопреки расхожему мнению, алгоритмы распознавания лиц могут работать и на системах с небольшой вычислительной мощностью.

Например, одноплатный микрокомпьютер Raspberry Pi 4B с производительностью старого андроид-смартфона.

Именно такой микрокомпьютер с 4 ГБ оперативной памяти и 4-ядерным процессором частотой 1,5 ГГц был использован в данном проекте.

4 ГБ

оперативная память

4 ядра

процессор

1,5 ГГц

тактовая частота

Программная часть: библиотеки

Библиотечная процедура face-recognition для Python 3 реализует программно распознавание лиц по алгоритму четырех квадратов dlib. Для отображения картинки с камеры необходимо установить библиотеку cv2. Серия команд для ОС имеет вид:

```
pip install face-recognition
pip install opencv-python
```

face-recognition
Распознавание лиц по алгоритму dlib

OpenCV (cv2)
Получение и отображение кадров камеры

Python 3
Среда выполнения программы

Тестирование ПАК: база данных ORL

Тестирование ПАК надо сделать с помощью БД ORL.

В ее состав входят изображения лиц с небольшими изменениями освещения, масштаба, пространственных поворотов, положения и различными эмоциями.

База представляет собой 400 фотографий 40 разных людей. Все фото представлены в градации серого.

400

фотографий в базе

40

разных людей

Градации серого

Формат всех фотографий

Фрагмент кода распознавания лица (1/2)

```
import face_recognition
import numpy as np
import cv2

# векторные представления лиц, имена, user_id в БД
known_face_encodings, known_face_names, known_face_uids = [], [], []

def run_rec():
    video_capture = cv2.VideoCapture(0, cv2.CAP_DSHOW)
    while video_capture.isOpened():
        ret, frame = video_capture.read()       # чтение кадра с камеры
        divideint = 2                           # уменьшение разрешения (для быстродействия)
        small_frame = cv2.resize(frame, (0, 0), fx=1/divideint, fy=1/divideint)
        # смена порядка каналов BGR → RGB, аналог [:, :, ::-1]
        rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)
```

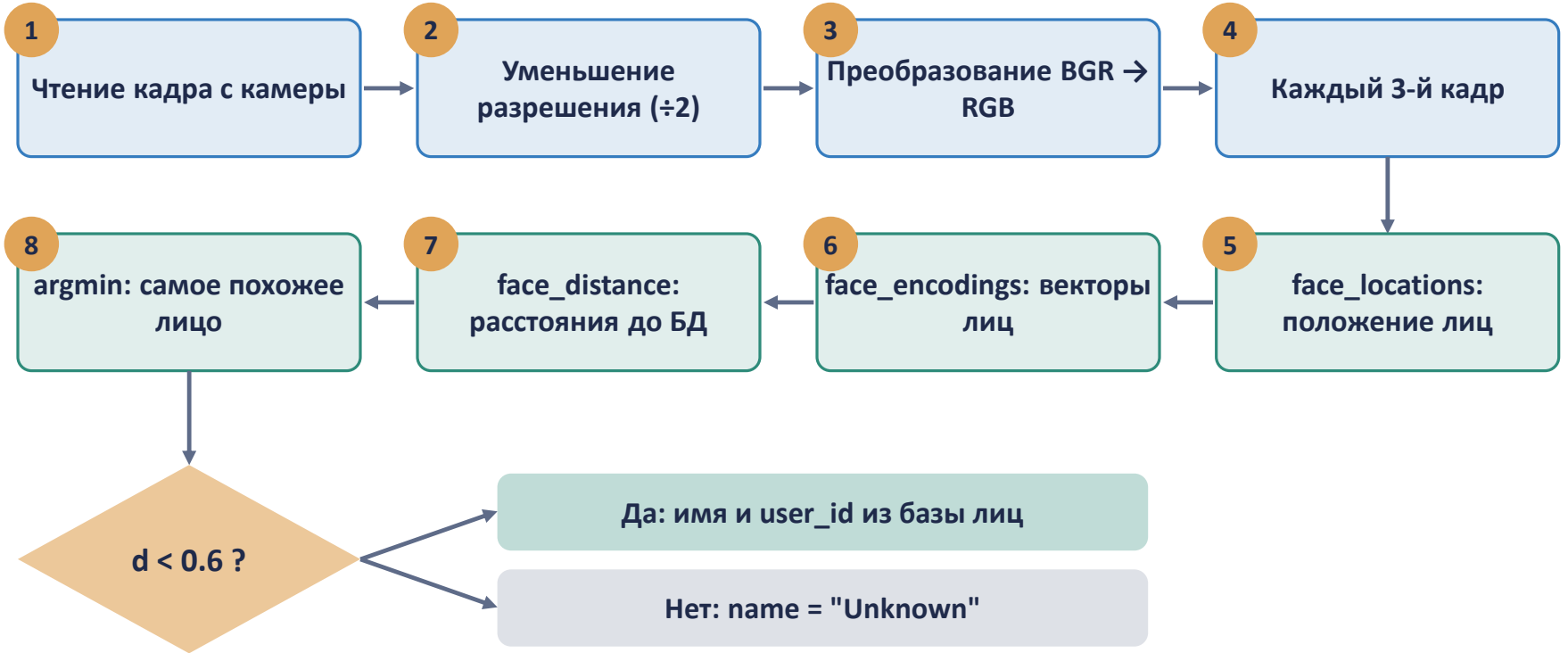
Рисунок 2. Подготовка: загрузка библиотек, структуры БД, захват и предварительная обработка кадра.

Фрагмент кода распознавания лица (2/2)

```
        # обрабатывается каждый 3-й кадр (для быстродействия)
        if process_this_frame % 3 == 0:
            # расположение лиц
            face_locations = face_recognition.face_locations(rgb_small_frame)
            # преобразование лиц в векторы
            face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
            # сравнение обнаруженных лиц с лицами в базе
            for face_encoding in face_encodings:
                name = "Unknown"
                # векторное расстояние — мера схожести векторов
                face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
                # индекс максимально похожего лица
                best_match_index = np.argmin(face_distances)
                # похожи, если векторное расстояние меньше 0.6
                if face_distances[best_match_index] < 0.6:
                    name = known_face_names[best_match_index]    # имя лица для пользователя
                    uid = known_face_uids[best_match_index]      # user_id для внутреннего использования
```

Рисунок 2. Распознавание: поиск лиц, векторизация, сравнение с базой по порогу 0,6.

Схема 2. Алгоритм распознавания лица



Распознавание лица и база данных

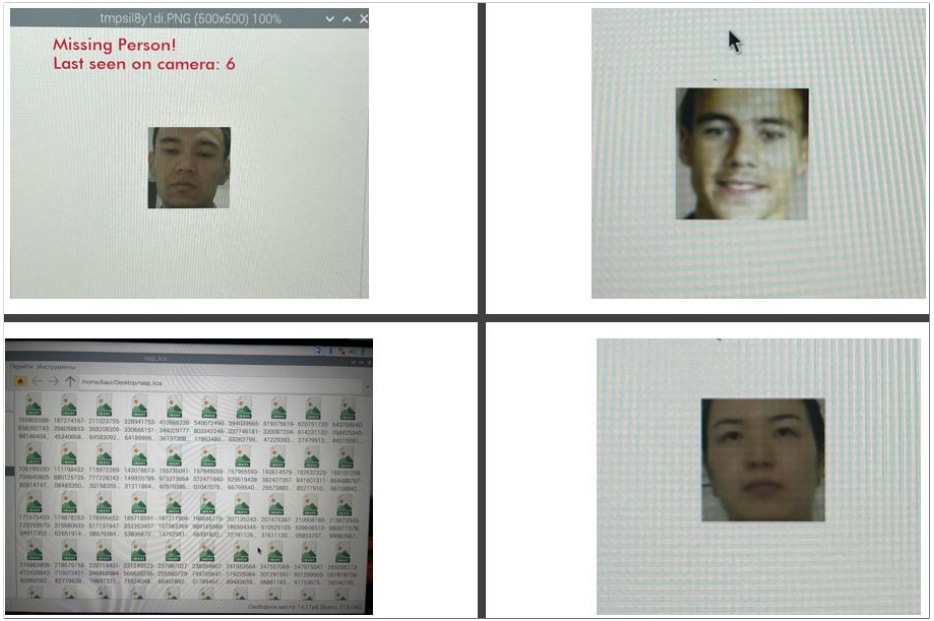


Рисунок 3 – Распознавание лица камерой прототипа и структура базы данных с лицами турфирмы

Ниже приведены изображения лиц и база данных в ОС Raspberry Pi.

Основным преимуществом базы данных о биометрии туристов является использование локального сервера, что решает ряд юридических проблем, связанных с утечкой личных данных в интернет.

Итоги

Предлагаемая простая и надежная система видеонаблюдения на базе мини-ПК является перспективным ПАК и имеет потенциал для расширенного использования.

В материале отрабатываются навыки программирования face-recognition для Python 3.

Для хранения БД с биометрией лиц клиентов турфирм просто и юридически грамотно использовать локальный сервер на одноплатном микрокомпьютере Raspberry Pi 4.

Перспективный ПАК

на базе мини-ПК

face-recognition

навыки программирования, Python 3

Локальный сервер

юридически грамотное хранение биометрии

Вопросы для самоконтроля

1. Опишите настройку библиотек для решения задачи распознавания лиц с помощью face-recognition для Python 3.

2. Дайте анализ юридических проблем из-за утечки биометрии клиентов через интернет.

3. Опишите состав ПАК системы видеонаблюдения на базе мини-ПК Raspberry Pi 4.

4. Обоснуйте, почему локальный сервер лучше защищает личные данные о биометрии по сравнению с базами данных в глобальных компьютерных сетях.