

КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ
ИМЕНИ АЛЬ-ФАРАБИ

ЛЕКЦИЯ № 7

Технологии распознавания лиц с использованием библиотеки face_recognition и их реализация на Raspberry Pi

Бельгибаев Бауржан Абдрахимович
Д.т.н., доцент · Кафедра «Искусственный интеллект и big data»
«Технологии в интеллектуальных системах управления в IoT» · 4 курс, бакалавриат

1 Цель лекции

Сформировать системное представление о распознавании лиц как задаче компьютерного зрения в интеллектуальных IoT-системах, рассмотреть математические основы идентификации и верификации, программную архитектуру face_recognition, особенности развёртывания на Raspberry Pi, методы экспериментальной оценки и ограничения применения биометрии.

2 План лекции

1 Распознавание лиц как подсистема интеллектуального управления

Интеллектуальная система управления (ИСУ) использует данные сенсоров, алгоритмы анализа и механизмы принятия решений для формирования управляющих воздействий. Камера является сенсором визуальной информации; распознавание лиц может выступать вспомогательной функцией контроля доступа, учёта присутствия или персонализации интерфейса. Решение о воздействии на исполнительное устройство должно приниматься отдельным контроллером с проверкой условий безопасности.

Необходимо различать обнаружение лица (определение области изображения), идентификацию (поиск наиболее близкого образца среди зарегистрированных лиц, задача 1:N) и верификацию (сравнение с заявленной личностью, задача 1:1). Эти задачи имеют разные критерии качества и различные риски ошибочного допуска.

Типовой поток данных: камера → получение кадра → предварительная обработка → детектор лица → выделение признаков → сравнение с шаблонами → логика принятия решения → журнал событий и интерфейс управления. В варианте Edge AI вычисления выполняются локально на Raspberry Pi, что уменьшает зависимость от облачного соединения, но не исключает требований к защите данных.

1 Распознавание лиц как подсистема интеллектуального управления (продолжение)

1.1. Структура IoT-решения

1 Распознавание лиц как подсистема интеллектуального управления (продолжение)

Компонент	Назначение	Пример
Визуальный сенсор	Получение кадров	USB-камера / Camera Module
Edge-узел	Обработка изображений	Raspberry Pi под Linux
Программный модуль	Детекция и извлечение признаков	Python, face_recognition, OpenCV
Хранилище	Эталонные признаки и метаданные	Локальная защищённая база
Контроллер	Логика допуска и безопасное управление	GPIO через согласующий модуль / PLC
Связь	События и телеметрия	MQTT с TLS / HTTPS

Математические основы распознавания

2.1. Цифровое изображение и детекция

Цветное изображение представляется тензором I ∈ R^(H×W×3), где H и W — высота и ширина кадра, а третья координата описывает цветовые каналы. Детектор возвращает прямоугольную область B=(x_min, y_min, x_max, y_max). При необходимости область нормализуется по масштабу и ориентации. Качество детекции зависит от освещения, угла поворота головы, перекрытий и разрешения.

2.2. Вектор признаков и расстояние

Модель извлечения признаков отображает изображение лица в вектор z=fθ(I_face) ∈ R^d. В типовом интерфейсе face_recognition применяется представление размерности d=128. Для двух векторов z_a и z_b вычисляется евклидово расстояние:

$$d(z_a,z_b) = \sqrt{[\Sigma(i=1...d)\ (z_ai - z_bi)^2]}.$$

Математические основы распознавания (продолжение)

Решение о совпадении принимается при $d(z_a,z_b) \le \tau$, где τ — порог, выбранный на валидационном наборе. Уменьшение τ обычно снижает ложные совпадения, но может увеличивать число отказов законным пользователям. Значение порога не является универсальным и требует настройки под условия эксплуатации.

2.3. Формирование решения

Для базы из N шаблонов $z_1,\dots,z_N$ определяется $j^*=\text{argmin}_j\, d(z,z_j)$. Если $d(z,z_j^*) \le \tau$, система возвращает метку j^*; иначе — «неизвестное лицо». Для более устойчивого решения можно агрегировать результаты нескольких последовательных кадров и требовать подтверждение в k из m наблюдений. Это повышает устойчивость к случайным ошибкам, но увеличивает задержку.

4 Конвейер распознавания лиц



Рис. 12. Этапы распознавания; порог τ задаёт баланс ложных принятий и отказов.

3 Библиотека face_recognition и программные зависимости

Python-библиотека face_recognition предоставляет функции обнаружения лиц, вычисления кодировок и сравнения признаков. Она использует модели dlib. В качестве вспомогательной библиотеки OpenCV может применяться для чтения видеопотока, преобразования цветового пространства, изменения размера кадра и отображения результата. Следует различать саму библиотеку face_recognition и пакет распознавания лиц как исследовательскую задачу: библиотека реализует готовый конвейер, но не устраняет необходимость проверки качества.

Основные функции: face_locations(image, model="hog"), face_encodings(image, known_face_locations), face_distance(known_encodings, candidate_encoding), compare_faces(known_encodings, candidate_encoding, tolerance=...). Режим HOG обычно более реалистичен для CPU Raspberry Pi; CNN-детектор может требовать значительно больше ресурсов. Доступность и совместимость пакетов зависят от модели Raspberry Pi, архитектуры ОС и версии Python.

3 Библиотека face_recognition и программные зависимости (продолжение)

3.1. Подготовка программной среды

Перед установкой следует обновить систему, создать виртуальное окружение Python и проверить доступность камеры. На Raspberry Pi OS современные версии системного Python рекомендуют установку сторонних библиотек в venv, а не в глобальную среду. Для dlib могут потребоваться компилятор, CMake и системные библиотеки. На маломощных моделях сборка может занимать продолжительное время; разумно использовать совместимые предварительно собранные пакеты, если они доступны из доверенного источника.

Пример последовательности команд:

```
sudo apt update
sudo apt install -y python3-venv python3-dev cmake build-essential
python3 -m venv ~/venvs/face-iot
source ~/venvs/face-iot/bin/activate
python -m pip install --upgrade pip
```

3 Библиотека face_recognition и программные зависимости (продолжение)

```
python -m pip install face_recognition opencv-python
```

Команды являются примером и могут потребовать адаптации. Если графические компоненты OpenCV не нужны, для серверного исполнения целесообразно рассмотреть headless-сборку. Проверка окружения: python -c "import cv2, face_recognition; print(cv2.__version__)".

4 Развёртывание на Raspberry Pi

Для учебного прототипа можно использовать Raspberry Pi 4 или 5, совместимую камеру, блок питания с достаточной мощностью, накопитель и охлаждение. Камера подключается через USB или интерфейс CSI с учётом используемого программного стека. На современных версиях Raspberry Pi OS камера CSI обычно работает через стек libcamera/Picamera2; прямой вызов cv2.VideoCapture(0) для неё не всегда подходит. Для USB-камеры VideoCapture часто достаточен.

Последовательность разработки: (1) проверить видеопоток; (2) получить согласованные эталонные изображения; (3) вычислить и безопасно сохранить кодировки; (4) реализовать сравнение в реальном времени; (5) настроить порог на независимых данных; (6) интегрировать события в систему управления; (7) испытать при разных условиях освещения и ракурсах.

4.1. Пример кода распознавания по USB-камере

4 Развёртывание на Raspberry Pi (продолжение)

```
import cv2
import face_recognition
reference = face_recognition.load_image_file("authorized.jpg")
reference_codes = face_recognition.face_encodings(reference)
if not reference_codes:
    raise RuntimeError("На эталонном фото лицо не обнаружено")
reference_code = reference_codes[0]
camera = cv2.VideoCapture(0)
if not camera.isOpened():
    raise RuntimeError("USB-камера недоступна")
try:
    while True:
        ok, frame = camera.read()
        if not ok:
            break
        small = cv2.resize(frame, (0, 0), fx=0.5, fy=0.5)
        rgb = cv2.cvtColor(small, cv2.COLOR_BGR2RGB)
```

4 Развёртывание на Raspberry Pi (продолжение)

```
        boxes = face_recognition.face_locations(rgb, model="hog")
        codes = face_recognition.face_encodings(rgb, boxes)
        for code in codes:
            distance = face_recognition.face_distance([reference_code], code)[0]
            label = "Совпадение" if distance <= 0.5 else "Неизвестно"
            print(f"{label}: расстояние={distance:.3f}")
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
finally:
    camera.release()
    cv2.destroyAllWindows()
```

Пример предназначен для учебной демонстрации, не реализует проверку «живости» и не должен непосредственно открывать физический замок. В промышленной системе требуется независимая логика безопасного допуска, журналирование, защита персональных данных и механизм отказа при недоступности вычислительного узла.

5 Организация эталонной базы

Для каждого участника экспериментального набора следует хранить идентификатор, версию модели, кодировку, дату согласия и метаданные качества изображения. Биометрические шаблоны являются чувствительными персональными данными: необходимы правовое основание обработки, ограничение доступа, шифрование, регламент хранения и удаления. По возможности следует не хранить исходные фотографии дольше необходимого.

При регистрации рекомендуется использовать несколько изображений с различным освещением и ракурсами, отделять регистрационные данные от тестовых и исключать утечки между выборками. Для нескольких шаблонов одного пользователя можно сравнивать входной вектор со всеми шаблонами, однако правило агрегации должно быть зафиксировано до испытаний.

Распределения расстояний (схематично)

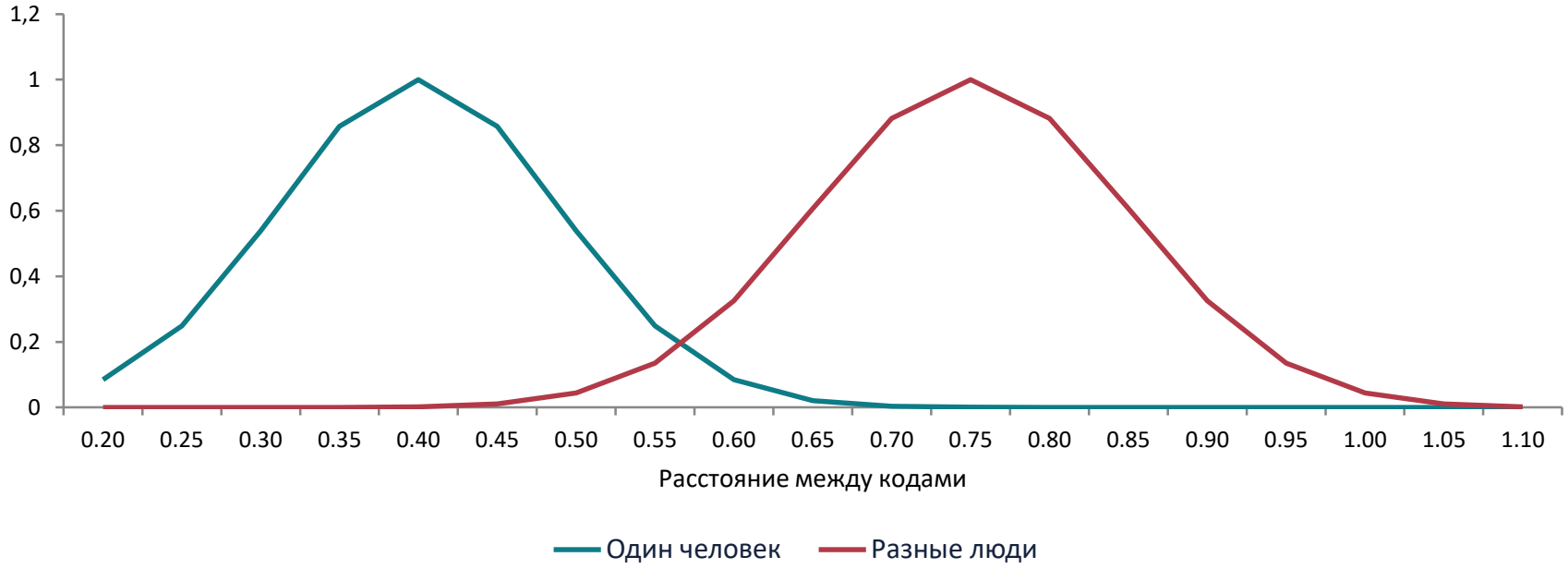

Рис. 13. Иллюстрация: пересечение распределений — причина ошибок; значения условные, не результат эксперимента.

6 Метрики качества и вычислительной эффективности

Для задачи верификации используются TP (верный допуск), TN (верный отказ), FP (ложный допуск), FN (ложный отказ). Точность положительных решений Precision=TP/(TP+FP), полнота Recall=TP/(TP+FN), F1=2·Precision·Recall/(Precision+Recall). При проверке биометрического допуска особенно важны FAR — доля ложных допусков и FRR — доля ложных отказов. Конкретные знаменатели зависят от принятого протокола испытаний; их необходимо указать в отчёте.

Вычислительные характеристики включают среднюю задержку T= T_capture + T_preprocess + T_detect + T_encode + T_match + T_action, частоту FPS≈1/T для последовательного конвейера, потребление оперативной памяти, загрузку CPU и среднюю мощность P. Энергия одного решения E=P·T. Для многопоточного конвейера FPS не обязательно равен обратной величине полной задержки.

6.1. Иллюстративный расчёт

6 Метрики качества и вычислительной эффективности (продолжение)

Предположим, что получение кадра занимает 30 мс, предварительная обработка — 15 мс, детекция — 140 мс, кодирование — 90 мс, сравнение — 5 мс, передача события — 20 мс. Полная задержка составляет 300 мс. Для строго последовательного конвейера предельная скорость составляет примерно 3,33 кадра/с. При средней мощности 7 Вт энергия одного решения $E=7 \cdot 0{,}300=2{,}1$ Дж. Это учебные исходные данные, а не результаты измерений Raspberry Pi.

6 Метрики качества и вычислительной эффективности (продолжение)

Этап	Условное время, мс
Получение кадра	30
Предобработка	15
Обнаружение	140
Извлечение признаков	90
Сравнение	5
Передача события	20
Всего	300

7 Экспериментальная методика

Исследование следует проводить на добровольно сформированном наборе данных с согласия участников. Рекомендуется выделить независимые наборы для регистрации, настройки порога и итогового тестирования. Для каждого режима измерить не менее нескольких десятков повторов на одинаковом оборудовании и зафиксировать версии ПО. Сценарии: дневное/искусственное освещение, фронтальный/боковой ракурс, разные расстояния, один/несколько объектов в кадре, низкая освещённость.

Для каждого сценария необходимо построить распределение расстояний для совпадающих и несовпадающих пар, график FAR–FRR в зависимости от порога, кривую ROC, а также диаграмму задержек и загрузки CPU. При сопоставлении платформ важно сохранять одинаковые изображения, версии моделей и протокол обработки. Результаты представлять с доверительными интервалами; на небольших выборках делать осторожные выводы.

8 Ограничения, этика и безопасность

Распознавание лиц не является безошибочным способом аутентификации. Ошибки возрастают при плохом освещении, частичном закрытии лица, изменении внешности и несоответствии условий тестирования условиям эксплуатации. Возможны атаки предъявления фотографий или видео. Для систем допуска необходимы отдельные механизмы обнаружения подделки, многофакторная проверка и безопасное состояние исполнительного оборудования.

В IoT-инфраструктуре необходимо защищать канал передачи событий (например, TLS), аутентифицировать устройства, ограничивать доступ к журналам и применять обновления безопасности. Использование биометрии должно соответствовать применимому законодательству и принципам минимизации данных; альтернативные способы доступа должны быть рассмотрены при проектировании.

9 Перспективы исследований

Научно значимыми направлениями являются адаптивный выбор порога в зависимости от качества изображения, исследование компромисса между точностью и задержкой, квантование моделей для Edge AI, распределённая обработка на Raspberry Pi и аппаратном ускорителе, а также проверка устойчивости к изменению домена данных. Важно отдельно оценивать, насколько прирост качества влияет на конечные показатели интеллектуальной системы управления, а не только на классификационную точность.

12 Заключение

Распознавание лиц в IoT является многоэтапной задачей, объединяющей получение изображений, извлечение признаков, принятие решений и интеграцию с контроллером. Raspberry Pi подходит для исследовательских прототипов, но допустимость применения определяется измеренными задержками, качеством распознавания, требованиями к безопасности и правовым режимом обработки биометрических данных. Использование библиотеки face_recognition ускоряет разработку, однако требует самостоятельной верификации на целевых сценариях.

13 Контрольные вопросы

1. В чём различие обнаружения, идентификации и верификации лица?
2. Какова роль Raspberry Pi в архитектуре Edge AI?
3. Какие этапы включает конвейер распознавания лиц?
4. Что означает вектор признаков и как он формируется?
5. Как вычисляется евклидово расстояние между кодировками?
6. Как выбор порога влияет на ложные допуски и ложные отказы?
7. Какие функции предоставляет библиотека face_recognition?
8. В чём особенности режимов HOG и CNN для маломощных устройств?
9. Какие проблемы могут возникнуть при установке dlib на Raspberry Pi?
10. Почему обработка CSI- и USB-камер может отличаться?
11. Какие данные следует хранить в базе эталонных шаблонов?

13 Контрольные вопросы (продолжение)

12. Как вычисляются Precision, Recall и F1?
13. Как измерить полную задержку распознавания и энергию одного решения?
14. Почему необходим независимый тестовый набор?
15. Какие угрозы возникают при биометрическом управлении физическим доступом?

14 Рекомендуемая литература и ресурсы

- **Перри Ли. Архитектура интернета вещей. М.:** ДМК Пресс, 2018.
- Матюшин А. О. Программирование микроконтроллеров: стратегия и тактика. М.: ДМК, 2016.
- **Raspberry Pi Documentation:** https://www.raspberrypi.com/documentation/
- **face_recognition:** https://github.com/ageitgey/face_recognition
- **dlib:** http://dlib.net/
- **OpenCV Documentation:** https://docs.opencv.org/
- **NIST Face Recognition Technology Evaluation:** https://pages.nist.gov/frvt/