КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ
ИМЕНИ АЛЬ-ФАРАБИ

ЛЕКЦИЯ № 9

Библиотеки компьютерного зрения с открытым исходным кодом. Применение OpenCV в интеллектуальных IoT-системах

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 методах цифровой обработки изображений и компьютерного зрения, архитектуре библиотеки OpenCV и способах интеграции алгоритмов визуального восприятия в контуры мониторинга и интеллектуального управления IoT-объектами.

2 Ожидаемые результаты

После изучения лекции докторант сможет: объяснять структуру конвейера компьютерного зрения; обоснованно выбирать методы фильтрации, сегментации и обнаружения объектов; разрабатывать прототипы на Python/OpenCV; оценивать точность, задержку и устойчивость обработки видеопотока; формулировать ограничения и план верификации Edge AI-системы.

3 План лекции

1. Компьютерное зрение в архитектуре IoT и интеллектуального управления
2. Открытые библиотеки: OpenCV и смежные инструменты
3. Представление изображений и предварительная обработка
4. Выделение признаков, сегментация, обнаружение и сопровождение объектов
5. Интеграция OpenCV с Raspberry Pi и нейроускорителями
6. Передача результатов в IoT и построение контура управления
7. Метрики качества, экспериментальная проверка и безопасность

1 Компьютерное зрение в интеллектуальных IoT-системах

Компьютерное зрение (computer vision) — совокупность алгоритмов получения измеримой информации из изображений и видеопоследовательностей. В интеллектуальных системах управления камера выступает датчиком, а результаты обработки превращаются в наблюдаемые переменные: координаты объекта, количество изделий, наличие дефекта, отклонение от траектории или признак опасной ситуации. В отличие от простого видеонаблюдения, такая система замыкает информационный контур: измерение → оценка состояния → принятие решения → управляющее воздействие → повторное измерение.

Типовая архитектура: камера (CSI/USB/IP) → получение кадров → коррекция и фильтрация → выделение признаков/инференс → логика принятия решений → MQTT/HTTP/Modbus-шлюз → контроллер/исполнительный механизм. Локальная обработка на Raspberry Pi уменьшает объём передаваемого видеотрафика и может снижать задержку по сравнению с облачной обработкой. Однако для критичных контуров управления необходимо отделять функции машинного зрения от независимых защитных блокировок.

Пример: камера над конвейером определяет наличие продукции в зоне контроля. Программа вычисляет бинарный признак присутствия, а ПЛК использует его для запуска сортировочного механизма. При пропуске кадров или потере связи система должна переходить в заранее определённое безопасное состояние.

2 Открытые библиотеки компьютерного зрения

OpenCV (Open Source Computer Vision Library) — библиотека компьютерного зрения и обработки изображений, предоставляющая функции захвата видео, преобразования цветовых пространств, фильтрации, морфологии, выделения контуров, геометрических преобразований, калибровки камер и запуска моделей глубокого обучения через модуль dnn. Основные интерфейсы — C++ и Python.

Смежные инструменты: NumPy — хранение и векторные операции над массивами; Pillow — базовые операции с изображениями; scikit-image — алгоритмы анализа и сегментации; PyTorch/TensorFlow — обучение нейронных моделей; ONNX Runtime — переносимый инференс совместимых моделей. Эти инструменты дополняют OpenCV, но не являются его обязательными компонентами.

2 Открытые библиотеки компьютерного зрения (продолжение)

Инструмент	Назначение	Роль в IoT
OpenCV	Кадры, геометрия, фильтрация, контуры	Визуальный сенсор и предобработка
NumPy	Числовые массивы и вычисления	Быстрые операции над матрицами
scikit-image	Сегментация, признаки, метрики	Исследовательская оценка алгоритмов
ONNX Runtime	Инференс обученных моделей	Развёртывание на периферии

3 Цифровое представление изображения

Цифровое изображение можно представить тензором I(x,y,c), где x и y — пространственные координаты, c — канал цвета. Для цветного кадра формата H×W×3 обычно используются 8-битные значения интенсивности. В OpenCV цветные кадры по умолчанию часто представлены в порядке BGR, что необходимо учитывать при преобразовании в RGB для сторонних библиотек.

Переход к оттенкам серого сокращает число каналов. Приближённая формула яркости для RGB: $Y = 0{,}299R + 0{,}587G + 0{,}114B$. Для устойчивого анализа применяются коррекция освещения, нормализация, размытие и подавление шума.

3.1. Фильтрация

Линейная пространственная фильтрация описывается свёрткой: $G(x,y) = \Sigma_i \Sigma_j K(i,j) \cdot I(x-i, y-j)$, где K — ядро фильтра. Гауссов фильтр сглаживает шум, но может ослаблять мелкие границы. Медианная фильтрация эффективна против импульсного шума. Выбор параметров фильтра должен учитывать масштаб полезных деталей и требования по задержке.

Цифровое представление изображения (продолжение)

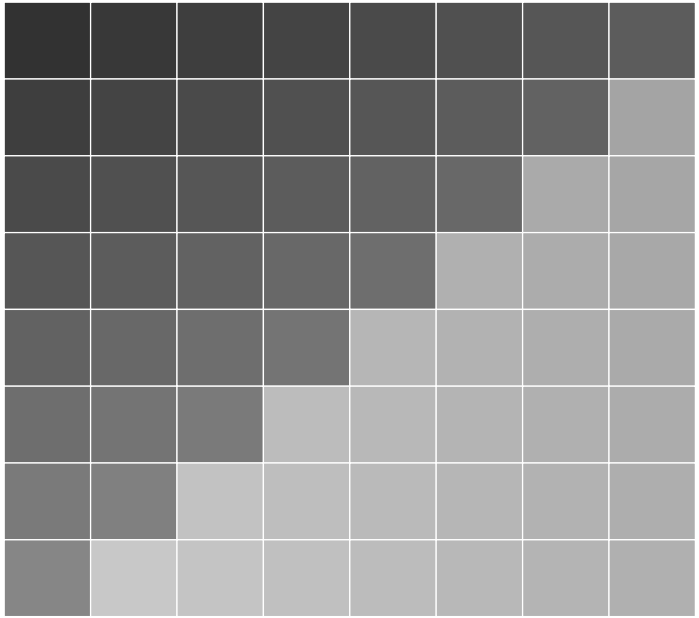
3.2. Бинаризация и морфология

Пороговая сегментация формирует маску $M(x,y)=1$, если $I(x,y)>T$, иначе $M(x,y)=0$. Для неравномерного освещения используют адаптивные пороги. Морфологические операции эрозии и дилатации с заданным структурным элементом позволяют удалять шумовые пиксели, закрывать разрывы и уточнять форму областей.

3.3. Выделение границ

Градиент интенсивности характеризуется компонентами $Gx=\partial I/\partial x$ и $Gy=\partial I/\partial y$; величина $|\nabla I|=\sqrt{(Gx^2+Gy^2)}$. Алгоритм Canny использует сглаживание, градиенты, подавление немаксимумов и двойную пороговую обработку. Он полезен для обнаружения геометрических границ при достаточном контрасте.

6 Цифровое изображение как матрица



8×8 пикселей, градации серого



I(x, y) ∈ [0, 255]; цветное — 3 канала

Рис. 16. Изображение — числовой массив; алгоритмы зрения работают с матрицами значений.

4 От признаков к обнаружению объектов

Контуры представляют границы связных областей и позволяют оценивать площадь, периметр, центр масс и ограничивающий прямоугольник. Для контура с моментами m_{00}, m_{10} и m_{01} центр определяется как $\bar{x}=m_{10}/m_{00}$, $\bar{y}=m_{01}/m_{00}$ при $m_{00}\neq 0$. Это полезно для управления захватом, позиционирования и подсчёта объектов.

Классические методы используют цветовые маски, оптический поток, ключевые точки и дескрипторы. Нейросетевые детекторы (например, семейства YOLO) возвращают классы, прямоугольники и оценки уверенности. OpenCV может выполнять предобработку и визуализацию, а сама модель может исполняться в OpenCV dnn или специализированном runtime. Совместимость конкретной модели и аппаратного ускорителя проверяется отдельно.

При сопровождении объектов важно сохранять идентичность объекта между кадрами. Для простой задачи допустима ассоциация по расстоянию между центрами; для сложных сцен применяют фильтр Калмана и алгоритмы сопоставления. Ошибки при перекрытии объектов приводят к двойному подсчёту, поэтому необходимо тестировать систему на реальных видеозаписях.

7 Конвейер обработки изображений OpenCV



Рис. 17. Типовая цепочка классической обработки; нейросетевые детекторы заменяют часть этапов.

5 Реализация на Raspberry Pi

Для практической работы требуется Raspberry Pi с поддерживаемой Raspberry Pi OS, камера CSI или USB, Python 3 и совместимая сборка OpenCV. Установка пакетов зависит от версии ОС и архитектуры; для системного варианта часто используется пакет python3-opencv из репозиториев ОС, а для изолированной среды — совместимый пакет opencv-python. Не следует смешивать системные и виртуальные окружения без проверки зависимостей.

5.1. Захват видеопотока и обнаружение движения

Следующий пример предназначен для USB-камеры, доступной как VideoCapture(0). Для камер Raspberry Pi Camera Module на новых версиях Raspberry Pi OS может потребоваться интерфейс Picamera2.

```
import cv2
cap = cv2.VideoCapture(0)
if not cap.isOpened():

raise RuntimeError("
Камера недоступна")
background = None
while True:
ok, frame = cap.read()
```

Реализация на Raspberry Pi (продолжение)

```
ask)

print("Пикселей движения:", moving_pixels)
```

Пример демонстрирует сравнение с фиксированным фоновым кадром и пригоден для контролируемой сцены. При изменении освещения или движении камеры количество «движущихся» пикселей будет ошибочно возрастать; для более сложных условий используют адаптивную модель фона, например MOG2, и дополнительную фильтрацию.

5.2. Передача события в IoT

Вместо пересылки полного видео узел может публиковать агрегированные признаки: наличие объекта, число обнаружений, время события и состояние камеры. MQTT подходит для телеметрии по модели публикации/подписки. В промышленном применении требуются аутентификация, шифрование, ограничения прав и контроль потери соединения.

5 Реализация на Raspberry Pi (продолжение)

```
import json
from datetime import datetime, timezone
#
payload отправляется через пре
дварительно настроенный MQTT-клиент
payload = {
    "timestamp": datetime.now(timezo
    "camera_id": "line_01",
    "object_detected": True,
    "c
    onfidence": 0.93
}
message = json.dumps(payload, ensure_ascii=False)
```

Поле confidence здесь демонстрационное: реальное значение должно поступать из алгоритма обнаружения, а не задаваться константой. Для принятия управляющего решения необходимо определить пороги, временную фильтрацию и реакцию на отсутствие данных.

6 Связь машинного зрения с системой управления

Рассмотрим дискретную систему $x(k+1)=Ax(k)+Bu(k)+w(k)$, $y(k)=Cx(k)+v(k)$, где x — состояние объекта, u — управление, y — измерение, w и v — возмущения и шумы. Камера предоставляет оценку измерения $\hat{y}(k)$, которая может поступать с задержкой τ. Если частота видеокадров низкая, а τ сопоставима с характерным временем объекта, устойчивость и точность управления могут ухудшиться.

Для визуального позиционирования зададим ошибку $e(k)=x_{target}-x_{detected}(k)$. Пропорциональное управление $u(k)=K_p \cdot e(k)$ может быть отправной точкой, но исполнительные ограничения, калибровка камеры, шум и задержки должны учитываться до физического запуска. Для сложных систем используются оценивание состояния, ПИД-регулирование, предиктивное управление и алгоритмы обучения с ограничениями безопасности.

Пример: мобильный робот движется вдоль размеченного коридора. Камера оценивает боковое отклонение e_y и угол курса e_ψ. Управляющая команда поворота может формироваться как $\omega=-k_y \cdot e_y - k_\psi \cdot e_\psi$. Коэффициенты выбираются экспериментально или методами синтеза регуляторов; при отсутствии достоверного распознавания линии движение должно быть ограничено.

7 Оценка качества и экспериментальная верификация

Показатели качества включают: точность обнаружения, precision=TP/(TP+FP), recall=TP/(TP+FN), F1=2PR/(P+R); среднее время обработки кадра, FPS≈1/t_frame, задержку от захвата до команды, загрузку CPU/RAM и энергопотребление. Для детекции объектов применяют IoU=|B_pred∩B_true|/|B_pred∪B_true| и mAP при заданных критериях сопоставления.

Важно разделять скорость модели и сквозную производительность: t_total=t_capture+t_preprocess+t_inference+t_postprocess+t_network+t_actuation. Даже быстрый инференс не гарантирует быстрого срабатывания исполнительного механизма. Измерения выполняются на целевом оборудовании при одинаковом разрешении и фиксированных условиях.

7.1. Пример расчёта

7 Оценка качества и экспериментальная верификация (продолжение)

Предположим, экспериментально получены времена: захват 18 мс, предобработка 12 мс, инференс 45 мс, постобработка 8 мс, передача 15 мс и исполнительная реакция 22 мс. Тогда сквозная задержка составляет 120 мс. Если оценивать только последовательную обработку до отправки результата (18+12+45+8=83 мс), теоретическая верхняя граница частоты составляет около 12 кадров/с без учёта перекрытия стадий и накладных расходов. Эти числа учебные, не паспортные показатели конкретной платформы.

7.2. План исследовательского эксперимента

Сформировать датасет из видеозаписей при дневном, искусственном и слабом освещении; разметить объекты; разделить данные по сценам на обучение, валидацию и независимый тест; сравнить минимум два метода; провести не менее 30 повторов измерений задержки для каждой конфигурации; представить среднее, медиану, 95-й перцентиль, доверительные интервалы и примеры типичных ошибок. Для задач управления отдельно оценить ложные срабатывания, пропуски и последствия отказов.

8 Кибербезопасность, приватность и ограничения

Камеры могут захватывать персональные данные. Необходимо минимизировать сохраняемую информацию, ограничивать доступ к видеопотоку и журналам, защищать каналы связи и определять сроки хранения. При использовании биометрического распознавания требуются отдельные правовые основания и оценка рисков. Интеллектуальный алгоритм не должен заменять аппаратные функции аварийной остановки в опасных производственных процессах.

Типичные ограничения: чувствительность к освещению, смаз изображения, недостаточная вычислительная мощность, перегрев устройства, несогласованность временных меток, ошибки калибровки и изменение распределения данных после внедрения. Их необходимо учитывать в техническом задании и протоколе испытаний.

12 Заключение

OpenCV обеспечивает базовые инструменты для превращения изображения в измеримую информацию и интеграции визуального восприятия с IoT-инфраструктурой. Практическая ценность решения определяется не только точностью алгоритма, но и устойчивостью при изменении условий, предсказуемостью задержек, безопасностью передачи данных и корректной связью с системой управления. Для докторантского исследования принципиально важны воспроизводимые эксперименты и независимая проверка на реальном объекте.

13 Контрольные вопросы

1. Какова роль компьютерного зрения в замкнутом контуре управления IoT-объектом?
2. Какие функциональные модули предоставляет OpenCV?
3. Чем отличаются BGR, RGB и оттенки серого?
4. Как математически описывается свёрточная фильтрация?
5. Когда следует применять медианный фильтр вместо гауссова?
6. В чём преимущества и ограничения пороговой сегментации?
7. Какие этапы включает алгоритм Canny?
8. Как вычислить центр масс бинарной области?
9. Чем классический детектор отличается от нейросетевого?
10. Какие особенности подключения камеры к Raspberry Pi следует учитывать?
11. Почему сквозная задержка важнее одного времени инференса?

13 Контрольные вопросы (продолжение)

12. Как организовать публикацию событий через MQTT?
13. Какие показатели характеризуют качество детектора?
14. Как спланировать независимую верификацию системы?
15. Какие меры безопасности обязательны при использовании видеокамеры в IoT?

Рекомендуемая литература и электронные ресурсы

1. OpenCV Documentation. https://docs.opencv.org/

2. Raspberry Pi Documentation. https://www.raspberrypi.com/documentation/

3. NumPy Documentation. https://numpy.org/doc/

4. scikit-image User Guide. https://scikit-image.org/docs/stable/

5. Eclipse Mosquitto Documentation. https://mosquitto.org/documentation/

6. Szeliski R. Computer Vision: Algorithms and Applications. 2nd ed. Springer, 2022.

7. Материалы силлабуса дисциплины «Технологии в интеллектуальных системах управления в IoT», 2026–2027.