

Федеральное государственное автономное
образовательное учреждение высшего образования
«Волгоградский государственный университет»

институт Математики и информационных технологий
кафедра Компьютерных наук и экспериментальной математики

Допустить работу к защите
Зав. каф. КНЭМ

_____ В.А.Клячин
« ____ » _____ 2021 г.

Курбанов Эльдар Ровшанович

**Система управления и формирования поведенческой стратегии
автономного мобильного робота**

Отчёт о научно-исследовательской работе

Студент

Курбанов Э.Р

(дата, подпись)

Специальность

Математическое обеспечение
и администрирование
информационных систем

Группа

МОСм-201

Научный
руководитель

д.ф.-м.н., зав. каф. КНЭМ
В.А.Клячин

(дата, подпись)

Нормокон-
тролер

(дата, подпись)

Волгоград 2021

Содержание

1. Введение	3
1.1. Цель работы	3
1.2. Задача распознавания лиц	3
2. Отечественное аппаратное решение	4
2.1. Спецификации ELISE-3D	4
2.1.1. Процессор ELISE	5
2.2. Программное обеспечение	7
2.3. Метод Виолы-Джонса	7
2.3.1. Принцип работы	7
2.3.2. Построение каскадов Хаара	8
2.3.3. OpenVX	9
2.4. Распознавание объекта	10
2.5. Результат	10
3. Зарубежное аппаратное решение	10
3.1. Спецификации Raspberry PI 3 Model B	11
3.2. Программное обеспечение	12
3.3. Дополнительная вычислительная мощность	12
3.4. Глубокие свёрточные нейросети	13
3.5. Инструментарий OpenVINO	14
3.5.1. Структура OpenVINO	14
3.5.2. Оптимизатор моделей нейронных сетей	15
3.5.3. Концепт механизма вывода	15
3.5.4. Препроцессинг моделей	16
3.6. Запуск распознавания лиц	16
3.6.1. Используемые модели нейронной сети	16
3.6.2. Код	17
3.6.3. Результат	22
4. Заключение	24

1. Введение

1.1. Цель работы

Целью данной работы является ознакомление читателя с некоторыми решениями задач по распознаванию лиц на встраиваемых системах. Из-за некоторых требований к встраиваемым системам, такие как мобильность, малая размерность, низкое энергопотребление в сочетании с низкой производительностью реализация решения данной задачи несколько усложняется и иногда нужно искать обходные пути.

В данной работе будет представлено решение задачи распознавания лиц на отечественном встраиваемом компьютере ELISE-3D и на одном из самых популярных зарубежных решений в области встраиваемых компьютеров – Raspberry PI 3 Model B.

1.2. Задача распознавания лиц

Задача автоматического распознавания лиц в реальном времени была поставлена ещё в прошлом веке, и сегодня она прекрасно реализована и используется во многих программно-аппаратных комплексах, в том числе и мобильных приложениях.

Технология распознавания лиц достигла невероятно высокого уровня: стало возможным распознавать конкретных людей, в том числе, в очках и головных уборах. Этот метод идентификации признан настолько точным, что он уже начал вытеснять обычные способы идентификации с помощью пароля, например, при совершении банковских платежей[1].

Однако мы не будем решать задачу идентификации личности по лицу. Разработка системы распознавания конкретных лиц требует специфичной выборки (тренировочных данных) для данной задачи, содержащей определённых лиц, которых мы и хотим распознавать. Образцы должны быть хорошими для качественного обнаружения. Создание такой выборки для конкретных людей - очень длительный и дорогой процесс (поскольку требуется много фотографий этих людей в разных ситуациях). При создании образца для простого распознавания любых лиц намного проще и быстрее, так как мы можем сделать большую учебную выборку, используя данные из сети Интернет.

На входящем видео-потоке мы распознаем лицо, попробуем угадать пол, возраст человека, а также его текущую эмоцию. Решение этой задачи - один из этапов проектирования будущего робота, который самостоятельно распознавать данные характеристики[21].

2. Отечественное аппаратное решение

Для решения задачи будем использовать отладочную плату ELISE-3D на базе процессора ELISE предоставленную Волгоградскому Государственному Университету компанией Элвис.

2.1. Спецификации ELISE-3D

ELISE-3D представляет собой стереокамеру со встроенной системой видео-аналитики. Модуль может использоваться для различных применений, требующих 3D анализ сцен и объектов, таких как системы безопасности, мобильные роботы и т.д. На модуле имеется 2 видеокамеры, выход USB для подключения Flash-накопителя с Buildroot, разъем Ethernet и вход питания на 12 вольт. Задняя часть ELISE-3D - это большой радиатор для отведения тепла. Фотографии можно найти на рисунке 1[23].

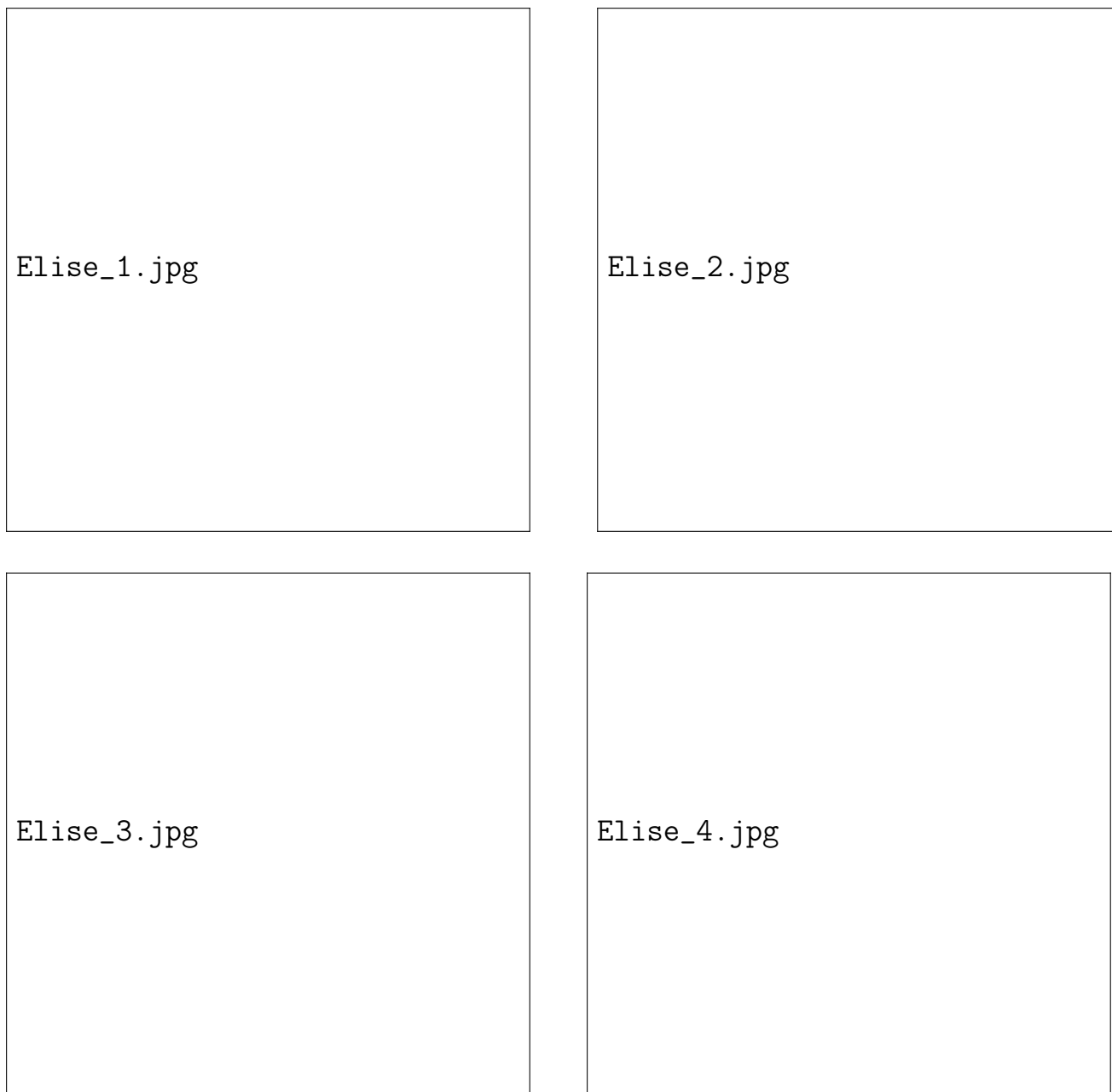


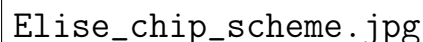
Рис. 1. ELISE-3D.

2.1.1. Процессор ELISE

Процессор ELISE - это мультиплатформенная система на кристалле с технологическим процессом 28 нм, ориентированный на интернет вещей и мультимедиа.

Система на кристалле ELISE объединяет широкий набор многофункциональных IP-блоков, включая блоки предварительной и последующей обработки стерео видеоизображений со сверхвысоким разрешением, несколько ядер центральных процессоров для выполнения задач разной степени интенсивности. Структуру можно увидеть на рисунке 2[21].

Основные спецификации процессора:



Elise_chip_scheme.jpg

Рис. 2. Структура системы на кристалле ELISE.

- Рабочая частота (0.9 В/ +85°C)
 - 1.2 ГГц Apache P5607 (CPU0)
 - 1.0 ГГц Intel Aptiv (CPU1)
 - 600 МГц MIPS M5150 Virtuoso (CPU2)
- Память на кристалле: Boot ROM 64 кбайт; System SRAM 128 кбайт; OTP 4 кбайт;
- Память, адресуемая микросхемой - DDR до 3.5 Гбайт
- Система управления питанием
- Встроенные датчики температуры и напряжения
- Процессор изображений PowerVR V2500 Felix, два потока
- Видеоэнкодер PowerVR E4500 Onyx, поддерживаемые форматы: H.264, MPEG-4, MPEG-2.
- Видеодекодер PowerVR D5500 Coral

Основной частью чипа является 8-ми ядерный DSP-кластер Elvees VELCore02, представляющий собой набор ядер для обработки видеосигнала. Предоставленные драйверы для данного устройства содержат интерфейсы для работы с OpenCL и OpenVX. OpenCV, к сожалению, отсутствует и использовать его здесь невозможно.

2.2. Программное обеспечение

ELISE-3D предоставляется с операционной системой на базе Buildroot. Это бесплатный набор Make-файлов и патчей с открытым исходным кодом, упрощающих и автоматизирующих процесс создания загружаемой среды Linux. Buildroot использует кросс-компиляцию, позволяющую ему создавать несколько целевых платформ в одной системе. Buildroot может автоматически собрать необходимый набор инструментов кросс-компиляции, создать корневую файловую систему, скомпилировать образ ядра Linux и сгенерировать загрузчик для целевой встроенной системы.

В первую очередь система предназначена для использования с небольшими или встроенными системами, основанными на различных компьютерных архитектурах. Множество стандартных библиотек C поддерживаются как часть набора инструментов, включая библиотеку GNU C, uClibc и musl, а также стандартные библиотеки C, которые принадлежат к различным предварительно настроенным средам разработки, таким как предоставляемые Linaro[22].

В сборку ELISE-3D вошло не так много программ. Из ключевого можно отметить присутствие свободной библиотеки компьютерного зрения OpenVX и интерпретатора Python версии 2.7.

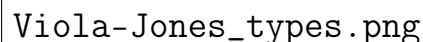
2.3. Метод Виолы-Джонса

Метод Виолы-Джонса - это первая инфраструктура обнаружения лиц, которая обеспечивает быстрое обнаружение лиц в режиме реального времени. Этот метод в 2001 году был предложен Полом Виолой и Майклом Джонсом[24].

2.3.1. Принцип работы

Признаки, используемые алгоритмом, опираются на суммирование пикселей из прямоугольных регионов. Сами признаки несколько напоминают признаки Хаара, которые ранее также использовались для поиска лиц на изображениях. На рисунке 3 показаны четыре различных типа признаков. Величина каждого вычисляется как сумма пикселей в белых прямоугольниках, из которой затем вычитается сумма пикселей в чёрных областях[25].

На этапе обнаружения в методе Виолы — Джонса окно установленного размера движется по изображению, и для каждой области изображения, над которой проходит окно, рассчитывается признак Хаара. Наличие или отсутствие предмета в окне определяется разницей между значением признака и обучаемым порогом. Поскольку признаки Хаара мало подходят для обучения или классификации (качество немного выше чем у случайной нормально распределенной величины), для описания объекта с достаточной точностью необходимо большее число признаков. Поэтому в методе Виолы — Джонса признаки Хаара



Viola-Jones_types.png

Рис. 3. Типы «признаков», использованные в алгоритме.

организованы в каскадный классификатор[26].

2.3.2. Построение каскадов Хаара

Для обучения каскада необходимо построить обучающую выборку в определённом формате. Алгоритм обучения каскада уже реализован в составе графического фреймворка OpenCV. Для обучения необходимо воспользоваться такими инструментами, как `opencv_traincascade` и `opencv_traincascade`.

Программа `Createsamples` подготавливает набор «хороших» изображений, приводя этот набор к общему формату путём выделения областей, на которых есть искомое лицо с учётом его пропорций на картинке. Программа `Traincascade` на основе полученных данных создаёт каскад из признаков Хаара, который и будет использоваться для распознавания лиц при помощи метода Виолы-Джонса.

Инструментам из OpenCV требуется обучающая выборка, записанная в определённом формате. Необходимо создать два списка с изображениями - «плохими» и «хорошими». Список «хороших» помимо пути к фотографии лиц должен содержать количество искомых лиц на изображении, а также координаты прямоугольников, в рамках которого искомые лица и содержатся[23].

На данный момент самым лучшим решением для создания такого специализированного формата списков является кроссплатформенный инструмент `GlmE3000`, созданный бывшим студентом Волгоградского Государственного Университета Вячеславом Чуприковым. Принцип работы программы очень прост: необходимо указать некоторую «рабочую» папку, в которой уже должны содержаться файлы `Good.dat` и `Bad.dat` (в самом начале они пустые), а также папки `Bad` и `Good` с соответствующими фотографиями внутри. Далее, в самой программе, работающей на оконном фреймворке `Swing` необходимо на каждой «хорошей» фотографии выделять искомое лицо или объект (программа универсальна). Таким образом после работы программы и оператора за

компьютером, выделяющего в прямоугольник лица мы получаем заполненные файлы Good.dat и Bad.dat, готовые для работы с инструментами библиотеки OpenCV. Интерфейс программы показан на рисунке 4.

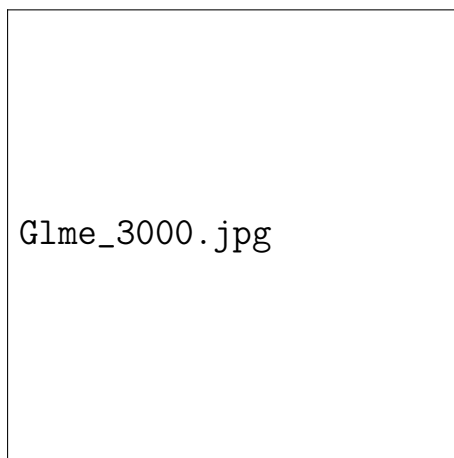


Рис. 4. Интерфейс программы G1mE3000.

2.3.3. OpenVX

OpenVX - это открытый, бесплатный лицензионный стандарт для кроссплатформенного ускорения приложений компьютерного зрения. Стандарт разработан Khronos Group для облегчения портативной, оптимизированной и энергоэффективной обработки методов для алгоритмов компьютерного зрения. OpenVX использует связанное графическое представление операций и является дополнением к библиотеке OpenCV с открытым исходным кодом. В некоторых приложениях платформа предлагает более оптимизированное управление графиками, чем OpenCV[27]. Структура распознавания лиц с двух видео-камер описана в рисунке 5[28].



Рис. 5. Распознавание на OpenVX с двух видео-потокков.

2.4. Распознавание объекта

Полноценного распознавания на данном аппарате в данный момент добиться не удалось. Программа распознавания лиц, предоставленная компанией Элвис, использующая метод Виолы-Джонса реализована не до конца в следствии проблем с производительностью самой отладочной платы и в следствии довольно сложного процесса разработки из-за особенностей Buildroot. Дело в том, что для тестирования какой-либо части программы необходимо заново скомпилировать весь Buildroot, затем, получив образ системы, закатать его на Flash-накопитель и только после этого протестировать. Такой подход не удобен, так как занимает довольно продолжительное время для решения даже самых маленьких задач.

В данный момент программа умеет в качестве аргумента принимать входное изображение в формате ppm и выдавать время обработки данного изображения. Пример результата работы программы можно увидеть на рисунке 6.

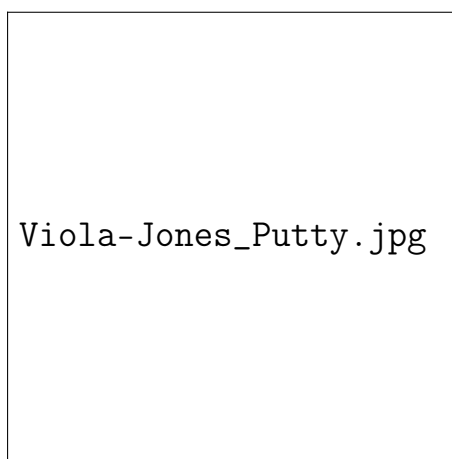


Рис. 6. Пример работы программы elcore40-viola-jones.

2.5. Результат

На данном этапе отечественному аппаратному решению требуется серьёзная доработка. В текущем виде оно не пригодно к использованию для распознавания лиц. К сожалению, компания Элвис «заморозила» проект. Основной причиной стала слабая производительность системы.

3. Зарубежное аппаратное решение

Поскольку нашей общей задачей является создание мобильного автономного робота с видеоаналитикой, мы предъявляем к оборудованию следующие требования:

- Мобильность;
- Компактность;
- Низкое энергопотребление.

Было принято решение использовать Raspberry Pi 3 Model B - одноплатный мини-компьютер, показанный на рисунке 7.

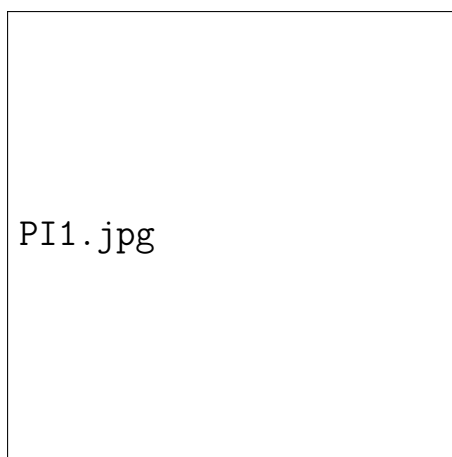


Рис. 7. Raspberry PI Model 3 Model B.

3.1. Спецификации Raspberry PI 3 Model B

Raspberry PI 3 Model B - это одноплатный компьютер размером с банковскую карту. Он предназначен для обучения программированию и способен выполнять самые различные задачи[4].

Характеристики:

- Четырехъядерный процессор 1,2 ГГц Broadcom BCM2837 64-битный процессор;
- 1 ГБ оперативной памяти;
- Наличие беспроводной локальной сети BCM43438 и Bluetooth Low Energy;
- Ethernet 100 мб/с;
- 40-контактный удлиненный GPIO;
- 4 USB версии 2.0;
- 4-х контактный стереовыход и композитный видеопорт;

- Полноразмерный HDMI для подключения видеовыхода;
- Порт камеры CSI для подключения камеры Raspberry Pi;
- Порт дисплея DSI для подключения сенсорного дисплея Raspberry Pi;
- Порт Micro SD для загрузки операционной системы и хранения данных;
- Модернизированный коммутируемый источник питания Micro USB до 2,5 А.

3.2. Программное обеспечение

Компьютер работает с MicroSD карты на свободной операционной системе Raspbian 9, основанной на Debian. В ней есть полноценный рабочий стол, можно запускать любые программы, скомпилированные под платформу Linux ARM. В репозиториях (apt-get) имеется большое количество программ доступных для скачивания и работы. Система использует менеджер пакетов dpkg[5].

3.3. Дополнительная вычислительная мощность

Распознавание лица в режиме реального времени на Raspberry Pi 3 Model B работает плохо, производя менее 1 кадра в секунду[6]. В связи с этим, мы будем вынуждены использовать дополнительные вычислительные мощности. В качестве дополнительного вычислительного устройства было решено использовать недавно (ноябрь 2018 г.) выпущенный Intel Neural Compute Stick 2, показанный на рисунке 8.

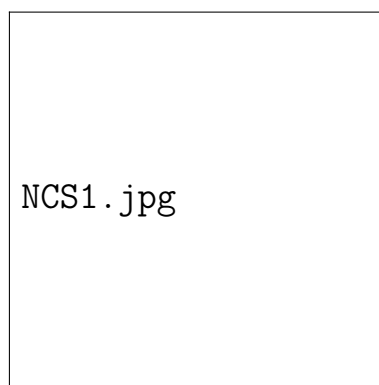


Рис. 8. Intel Neural Compute Stick 2.

3.4. Глубокие свёрточные нейросети

Задачу распознавания будем решать с помощью глубоких многослойных нейронных сетей. В слоях такой сети сама свёртка чередуется с субдискретизацией (пулингом). Такая архитектура позволяет распознавать признаки со сложной иерархией. Структура таких сетей всегда многослойна, однонаправлена, слои не имеют обратной связи. При обучении нейронной сети используется метод обратного распространения ошибки. Архитектура типичной свёрточной нейронной сети показана на рисунке 9[2].

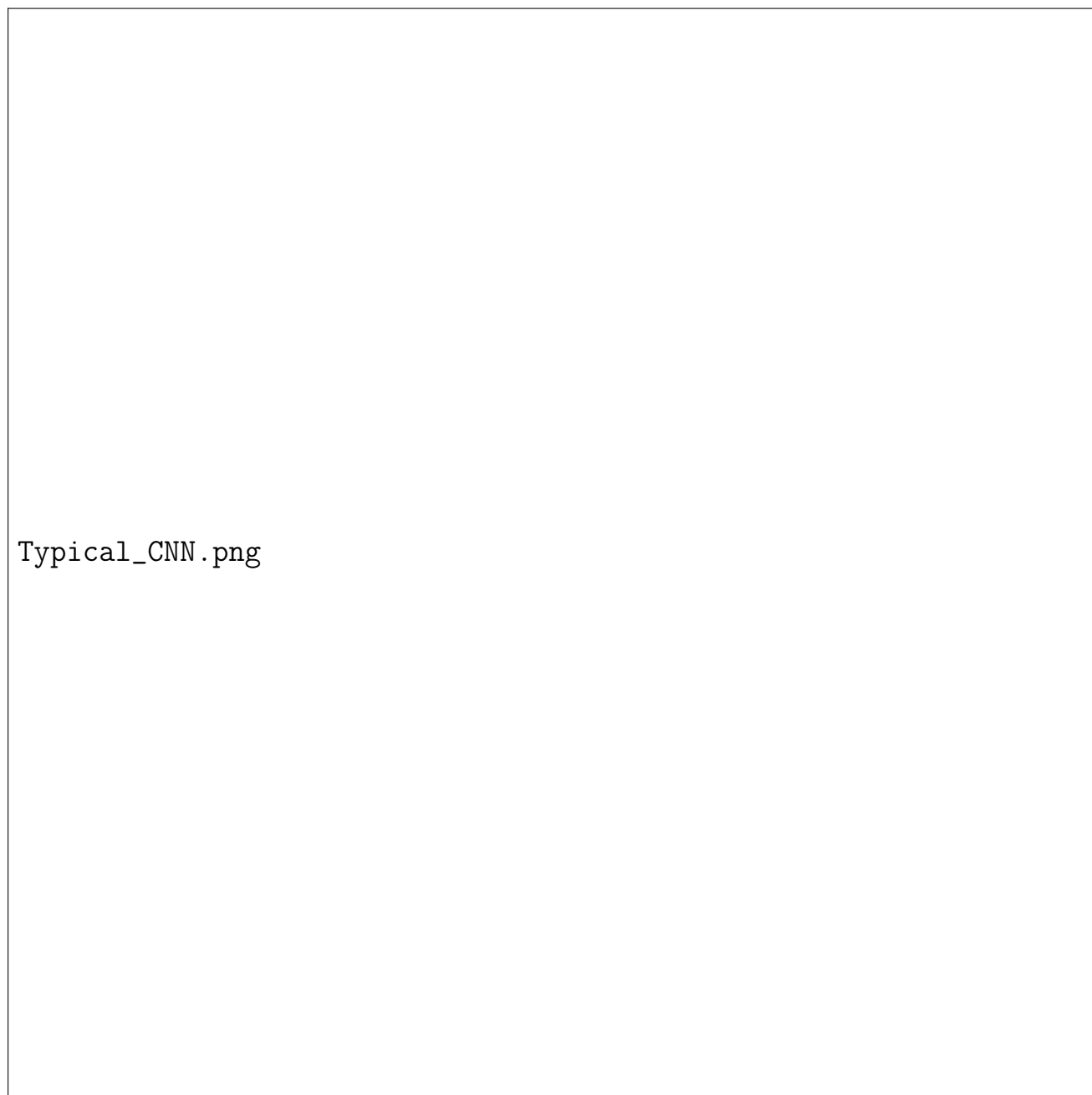


Рис. 9. Типовая архитектура свёрточной нейронной сети.

3.5. Инструментарий OpenVINO

OpenVINO - это инструментарий для разработки программного обеспечения под платформу Intel, и его аббревиатура означает - открытый визуальный вывод и оптимизатор нейронной сети. Он содержит инструментарий для развёртывания нейронных сетей глубокого обучения для различных устройств Intel, будь то процессор, встроенная графика, вентиляционная матрица, или процессоре машинного зрения[7].

3.5.1. Структура OpenVINO

Набор инструментов OpenVINO состоит из нескольких компонентов: (также показано на рисунке 10)

Intel Deep Learning Deployment Toolkit - это набор инструментов для разработчиков, который позволяет развёртывать предварительно обученные углубленные модели обучения через интерфейс высокого уровня. Инструментарий поддерживает выполнение на четырёх типах устройств:

- Графический процессор (GPU) - это специализированная электронная схема, предназначенная для быстрой манипуляции с памятью для ускорения создания изображений в буфере кадров, предназначенном для вывода на устройство отображения[8].
- Центральный процессор (CPU), также называемый центральным или главным процессором, представляет собой электронную схему внутри компьютера, которая выполняет инструкции компьютерной программы, выполняя основную арифметику, логику, управляющие и входные / выходные (I/O) операции задающимися инструкциями[9].
- Процессор машинного зрения (VPU) является новым классом микропроцессоров; это особый тип ускорителя искусственного интеллекта, предназначенный для ускорения выполнения задач машинного зрения[10].
- Программируемая пользователем вентиляционная матрица (FPGA) - это интегральная схема, разработанная для конфигурирования пользователем или инженером непосредственно после изготовления - по этой причине возник термин «программируемый на месте»[11].

При работе с компьютерным зрением инструментарий OpenVINO нацелена на использовании OpenCV и OpenVX. Набор инструментов OpenVINO также включает в себя большой набор предварительно обученных моделей с глубоким обучением.

Мы использовали Intel Neural Compute Stick 2 на основе графического процессора Intel Movidius Myriad X, который принадлежит к классу процессоров

машинного зрения. Это оборудование, OpenVINO и встроенное в VPU программное обеспечение предназначены для механизма вывода нейронной сети в реальном времени.

3.5.2. Оптимизатор моделей нейронных сетей

Чтобы успешно использовать предварительно глубокую обученную нейронную сеть, необходимо преобразовать ее из привычного формата, используемого в средах глубокого обучения, таких как TensorFlow, Caffe, ONNX в оптимизированную модель IR, которая состоит из файлов формата BIN и XML. Для такого преобразования можно использовать включенную в комплект программу Model Optimizer. Типовой рабочий процесс развертывания обученной модели глубокого обучения показан на рисунке 11[12].

3.5.3. Концепт механизма вывода

Нам нужно взять предварительно обученную модель и подготовить ее к выводу с помощью специального программного обеспечения - оптимизатора модели от OpenVINO.

Оптимизатор выполняет различные виды манипуляций с моделями, оптимизирует их и преобразует в IR-формат. Все эти манипуляции не зависят от аппаратного обеспечения, но результат OpenVINO может быть воспроизведен на различных устройствах[13].

Inference Engine (механизм вывода) - это некоторое API, предоставляющее вывод оптимизированных предварительно обученных нейронных сетей, работающих в реальном времени. Оно является общим для всех аппаратных устройств и поддерживает такие языки программирования, как C++ и Python.

Механизм вывода находится в прошивке самого MYRIAD (процессор машинного зрения).

Таким образом, логический механизм обработки имеет разные плагины для каждого из поддерживаемых устройств. Когда разработчик приложения пользуется этим API, вызывая соответствующие функции, он должен указать устройство, на котором должна работать его модель нейронной сети. Механизм логического вывода будет выбирать необходимый плагин и, следовательно, использовать правильную аппаратную реализацию функций, необходимых для работы нейронной сети (например, умножение на центральном процессоре выполняется при помощи команды ассемблера, а на видеокарте для этого уже понадобится команда из OpenCL).

Наше приложение, распознающее лица берёт кадр с видеокамеры и отправляет его в Inference Engine Common API. Механизм Вывода Intel подаёт на вход нейронной сети полученный кадр и на выходе мы получаем параметры,

которые мы используем для отрисовки распознанных образов. Структура Механизма Вывода Intel показана на рисунке 12[14].

3.5.4. Препроцессинг моделей

Процесс работы с использованием плагина Inference Engine на нашем устройстве (MYRIAD) выглядит так, как показано на рисунке 13.

Перед отправкой нейронной сети определенному плагину, который принимает оптимизированные нейронные сети в качестве входа от оптимизатора моделей, она проходит несколько этапов оптимизации:

- 1) Оптимизация на уровне нейронной сети: это включает в себя отображение операций, используемых в нейронной сети к архитектуре конкретного целевого устройства для запуска. Выполняя этот шаг заранее, Механизм Вывода улучшает производительность и минимизирует время обработки, так как этот шаг не придётся делать во время непосредственной работы нейронной сети.
- 2) Оптимизация на уровне памяти: этот шаг заключается в переупорядочивании данных, размещаемых в памяти для соответствия некоторым особым требованиям к целевым устройствам.
- 3) Оптимизация на уровне ядра. Механизм Вывода выбирает правильную реализацию команд, которая лучше всего подходит для данного набора инструкций целевой архитектуры.

3.6. Запуск распознавания лиц

Общая схема работы приложения изображена на рисунке 14[3].

3.6.1. Используемые модели нейронной сети

Чтобы продемонстрировать преимущества Neural Compute Stick 2 в разрабатываемом мобильном роботе, мы использовали предварительно обученные модели, которые могут работать параллельно в механизме логического вывода, такие как: face-detection-adas-0001, age-gender-recognition-retail-0013, head-pose-estimation-adas-0001, emotions-recognition-retail-0003.

- Модель распознавания лиц (face-detection-adas-0001) является основной моделью. Она позволяет идентифицировать лица на изображении, включает в себя данные глубокую обученную нейронную сеть, позволяющая выполнить объем вычислений для свёрточного блока 3x3. Эта сеть имеет 252 слоя[15].

- Модель распознавания пола и возраста (age-gender-recognition-retail-0013) - это свёрточная нейронная сеть для одновременного распознавания возраста и пола. Сеть может распознавать возраст людей в диапазоне [18, 75] лет и имеет 32 слоя[16].
- Модель оценки позы головы (head-pose-estimation-adas-0001) - это сеть оценки состояния головы, основанная на архитектуре глубокой свёрточной нейронной сети. Угловые регрессионные слои представляют собой свёртки, а нормы пакетов полностью связаны с одним выходом. Эта сеть имеет 43 слоя[17].
- Модель распознавания эмоций (emotions-recognition-retail-0003) представляет собой полностью свёрточную сеть для распознавания пяти эмоций («нейтральная», «счастливая», «грустная», «удивительная», «гнев»). Сеть имеет 54 слоя[18].

3.6.2. Код

Для обработки изображений в реальном времени нам потребуется использовать открытую библиотеку компьютерного зрения OpenCV. VideoCapture - это класс C++, предназначенный для захвата кадров с камеры или потокового видео[19].

```
1 cv::VideoCapture cap;
```

Далее, мы пытаемся открыть поток, чтобы прочитать первый кадр из камеры и если произошла ошибка или кадр ничего не содержит, то прерываем программу.

```
1 throw std::logic_error("Произошла ошибка при открытии потока с  
→ камеры");
```

Первым шагом программы будет загрузка аппаратного плагина MYRIAD в Inference Engine. Как упомянуто выше - это необходимый шаг для правильной обработки нейронной сети на нужном устройстве. В нашем случае это Intel Neural Compute Stick 2 (с кодовым названием MYRIAD).

PluginDispatcher - это класс C++, который выбирает нужный плагин, то есть набор API для механизма вывода, чтобы получить результаты работы нейронных сетей на конкретном аппаратном обеспечении[20].

```
1 //Получаем плагин по имени из Inference Engine класса  
→ PluginDispatcher.  
2 InferenceEngine::InferencePlugin plugin =  
→ InferenceEngine::PluginDispatcher().getPluginByDevice("MYRIAD");
```

Вторым шагом станет чтение IR моделей и загрузка их в MYRIAD:

```
1 InferenceEngine::CNNNetReader netReader;
2 // Загружаем файл, описывающий топологию нейронной сети.
3 netReader.ReadNetwork("face-detection-adas-0001.xml");
4 // Загружаем файл, описывающий веса для нейронной сети.
5 netReader.ReadWeights("face-detection-adas-0001.bin");
6 // Получаем готовую глубокую свёрточную нейронную сеть.
7 InferenceEngine::CNNNetwork cnNet = netReader.getNetwork();
8 // Загружаем нейронную сеть в плагин и готовим её к работе в
   ↳ присоединённом устройстве MYRIAD.
9 InferenceEngine::ExecutableNetwork net = plugin.LoadNetwork(cnNet);
10
11 // Повторяем эти действия для другой нейронной сети, определяющей
   ↳ положение головы в пространстве.
12 InferenceEngine::CNNNetReader netReaderHeadPose;
13 // Загружаем файл, описывающий топологию нейронной сети.
14 netReaderHeadPose.ReadNetwork("head-pose-estimation-adas-0001.xml");
15 // Загружаем файл, описывающий веса для нейронной сети.
16 netReaderHeadPose.ReadWeights("head-pose-estimation-adas-0001.bin");
17 // Получаем готовую глубокую свёрточную нейронную сеть.
18 InferenceEngine::CNNNetwork cnNetHeadPose =
   ↳ netReaderHeadPose.getNetwork();
19 // Загружаем нейронную сеть в плагин и готовим её к работе в
   ↳ присоединённом устройстве MYRIAD.
20 InferenceEngine::ExecutableNetwork netHeadPose =
   ↳ plugin.LoadNetwork(cnNetHeadPose);
21
22 // Похожие действия производим для моделей
   ↳ age-gender-recognition-retail-0013 и
   ↳ emotions-recognition-retail-0003.
```

Третьим шагом станет выполнения бесконечного цикла, в котором производим вычисления и показываем результат пользователю на экран:

```
1 // В этой структуре храним результаты распознавания лиц.
2 struct Result {
3     // Вероятность того, что лицо распознано верно.
4     float confidence;
5     // Позиция лица, представленная прямоугольной рамкой.
6     cv::Rect location;
7 }
```

```

8
9 // В этой структуре храним результаты распознавания позы головы.
10 struct ResultsHeadPose {
11     float angle r;
12     float angle p;
13     float angle y;
14 };
15
16 // В этой структуре храним указатель на тип структуры, конструктор
17     ↪ и определяем методы для работы с различными данными.
18 struct Face {
19     using Ptr = std::shared_ptr<Face>;
20     explicit Face(size_t id, cv::Rect& location); // Принимает
21     ↪ id и местоположение.
22     void updateHeadPose(struct ResultsHeadPose values); //
23     ↪ Принимает значения положения головы.
24 }
25
26 // Массивы структур для хранения данных.
27 std::vector<Result> results;
28 std::vector<ResultsHeadPose> resultsHeadPose;
29
30 // Основной и бесконечный цикл приложения. Каждая итерация данного
31     ↪ цикла определяет кадр с результатами вычислений.
32 while (true) {
33
34     // Очищаем до этого распознанные лица.
35     results.clear();
36     // Объявляем указатель на будущий кадр.
37     cv::Mat *frame;
38     // Из VideoCapture получаем следующий кадр, а результат
39     ↪ записываем в frame.
40     cap.read(frame);
41
42     // Создаём запрос в Inference Engine.
43     InferenceEngine::InferRequest::Ptr *request =
44     ↪ net.CreateInferRequestPtr();
45     // Из сформированного запроса получаем ссылку на объект, в
46     ↪ который сложим наши входные данные - кадр из камеры.
47     Blob::Ptr inputBlob = request->GetBlob();

```

```

41 // Из предыдущего объекта получаем указатель на буффер, в
   ↳ который и сложим выходные данные, затем преобразуем
   ↳ буффер к нужному формату - указатель на float.
42 float* detections = inputBlob.buffer().as<float*>();
43 // Вызываем функцию, принимающую кадр и указатель на объект
   ↳ Blob. После выполнения функции в буфере появятся данные
   ↳ о распознавании и их можно начать вытаскивать и
   ↳ обрабатывать.
44 matU8ToBlob<uint8_t>(frame , &inputBlob);
45
46 // Данные получены. Начинаем обрабатывать.
47 for (int i = 0; i < max; i++) {
48
49     // Создаём структуру и складываем туда информацию о
   ↳ найденных лицах. Нам пригодится местоположение
   ↳ лица на кадре и вероятность правильного
   ↳ распознавания.
50     Result result;
51     result.confidence = detections[i * objectSize + 2];
52     result.location.x = static cast<int>(detections[i *
   ↳ objectSize + 3] * width);
53     result.location.y = static cast<int>(detections[i *
   ↳ objectSize + 4] * height);
54     result.location.width = static
   ↳ cast<int>(detections[i * objectSize + 5] *
   ↳ width - result.location.x);
55     result.location.height = static
   ↳ cast<int>(detections[i * objectSize + 6] *
   ↳ height - result.location.y);
56
57     //Если вероятность слишком мала, то лицо не
   ↳ добавляем и идём на следующую итерацию смотреть
   ↳ следующие объекты, похожие на лица.
58     if (result.confidence <= threshold )
59         continue;
60     // В другом случае добавляем.
61     results.push_back(result);
62
63     //Лицо распознали. Теперь таким же образом получим
   ↳ положение головы, воспользовавшись
   ↳ соответствующей нейронной сетью, запущенной на
   ↳ том же устройстве MYRIAD X.

```

```

64 // Создаём запрос в Inference Engine.
65 request = netHeadPose.CreateInferRequestPtr();
66
67 // Из сформированного запроса получаем ссылки на
68   ↳ объекты, в который сложим наши входные данные -
69   ↳ кадр из камеры. Каждая ссылка соответствует
70   ↳ своей координате. Далее из этих ссылок просто
71   ↳ вытащим число и запишем в результат.
68 Blob::Ptr angleR = request->GetBlob("angle r fc");
69 Blob::Ptr angleP = request->GetBlob("angle p fc");
70 Blob::Ptr angleY = request->GetBlob("angle y fc");
71 ResultsHeadPose resultHeadPose = {
72     angleR->buffer().as<float*>()[i],
73     angleP->buffer().as<float*>()[i],
74     angleY->buffer().as<float*>()[i]
75 };
76 resultsHeadPose.push_back(resultHeadPose);
77
78 // Похожие действия повторяем и для остальных
79   ↳ нейронных сетей.
79 <...>
80 }
81
82 //Счётчик лиц, присваивающий каждому уникальный номер.
83 size_t id = 0;
84
85 // Формируем конечные данные и показываем на кадре.
86 for(int i = 0; i < results.size(); i++) {
87
88     // Получаем объект класса, описывающий лица.
89     Face::Ptr face = std::make_shared<Face>(id++,
90   ↳ rect);
91     // И передаём туда положение головы и прочие
92   ↳ распознанные ранее параметры.
91 face->updateHeadPose(resultsHeadPose[i]);
92 <...>
93
94 // В окне пользовательского вывода рисуем
95   ↳ прямоугольник вокруг лица.
95 cv::rectangle(frame, rect, cv::Scalar(255 , 0 , 0),
96   ↳ 1);

```

```

96      // Над лицом отображаем извлечённую из всех
97      ↪ нейронных сетей информацию в виде текста.
cv::putText(frame, face->sex + face->age +
98      ↪ face->emotion,
99      ↪ cv::Point2f(results[i].location.x ,
100      ↪ results[i].location.y),
101      ↪ cv::FONT_HERSHEY_COMPLEX_SMALL, 1.5, cv::Scalar
102      ↪ (255, 0, 0), 2);
103
104      // Находим центр лица для отрисовки в центре
105      ↪ визуализацию текущего положения головы.
cv::Point3f center(results[i].location.x +
106      ↪ results[i].location.width / 2,
107      ↪ results[i].location.y +
108      ↪ results[i].location.height / 2, 0.0f);
109      // Отрисовку визуализации текущего положения головы
110      ↪ отдаём в другую большую функцию.
drawHeadPose(frame, center, resultsHeadPose[i]);
111
112     }
113 }

```

3.6.3. Результат

На рисунке 15 можно видеть результат работы программы.



Рис. 15. Примеры работы программы.

При решении данной задачи достигнута отметка 18 кадров в секунду. Для достижения наиболее лучших результатов мы можем использовать два или более Intel Neural Compute Stick 2.

FPS недостаточно велик, однако, учитывая количество обнаруживаемых параметров, это нормальный результат. На производительность также влияет визуализация положения головы в пространстве - это достаточно сложная задача для механизма вывода. Если мы исключим некоторые характеристики, то можно сказать, что Raspberry Pi 3 Model B и Intel Neural Compute Stick 2 могут обнаруживать лица в режиме реального времени.

4. Заключение

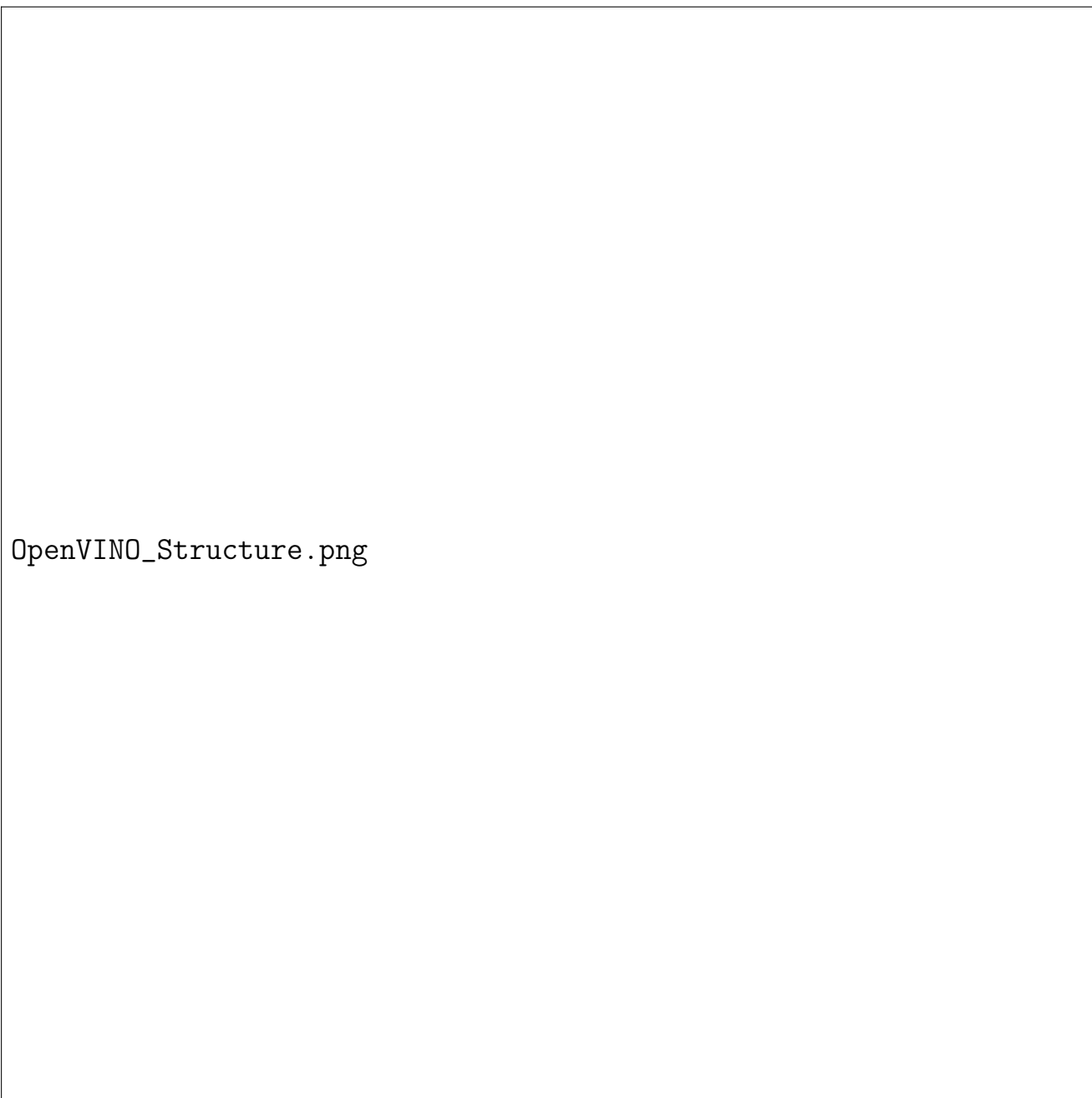
Подводя итоги работы, можно сказать, что отечественные аппаратные решения от компании Элвис в области компьютерного зрения на данный момент находятся в «плачевном» состоянии и пока что для разработки нашего мобильного робота мы вынуждены использовать зарубежную разработку от Raspberry Foundation и компании Intel, а такое партнёрство с зарубежными коллегами при попытках внедрения данных решений в государственной сфере на вряд-ли будет одобрено. Однако, это ни сколь не помешало решению нашей задачи и наш робот сможет распознавать лица.

Список литературы

- [1] People's Daily Online. Is facial recognition the future of smart payment in China? — The Telegraph. <https://www.telegraph.co.uk/peoples-daily-online/science/facial-recognition/>
- [2] Свёрточная нейронная сеть. — Википедия. https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%91%D1%80%D1%82%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C
- [3] Integrate the Inference Engine New Request API with Your Application. — Intel. https://docs.openvinotoolkit.org/latest/_docs_IE_DG_Integrate_with_customer_application_new_API.html
- [4] Raspberry Pi 3 Model B. — RASPBERRY PI FOUNDATION. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>
- [5] Raspbian. — Википедия. <https://ru.wikipedia.org/wiki/Raspbian>
- [6] Adrian Rosenbrock. Raspberry Pi Face Recognition. — PyImageSearch. <https://pyimagesearch.com/2018/06/25/raspberry-pi-face-recognition/>
- [7] Intel Distribution of OpenVINO Toolkit. — Intel. <https://software.intel.com/ru-ru/openvino-toolkit>
- [8] Графический процессор. — Википедия. https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80
- [9] Центральный процессор. — Википедия. https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D1%82%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80
- [10] Процессор машинного зрения. — Википедия. https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D1%82%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80
- [11] Программируемая пользователем вентильная матрица. — Википедия. https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D0%B5%D0%BC_%D0%B2%D0%B5%D0%BD%D1%82%D0%B8%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0

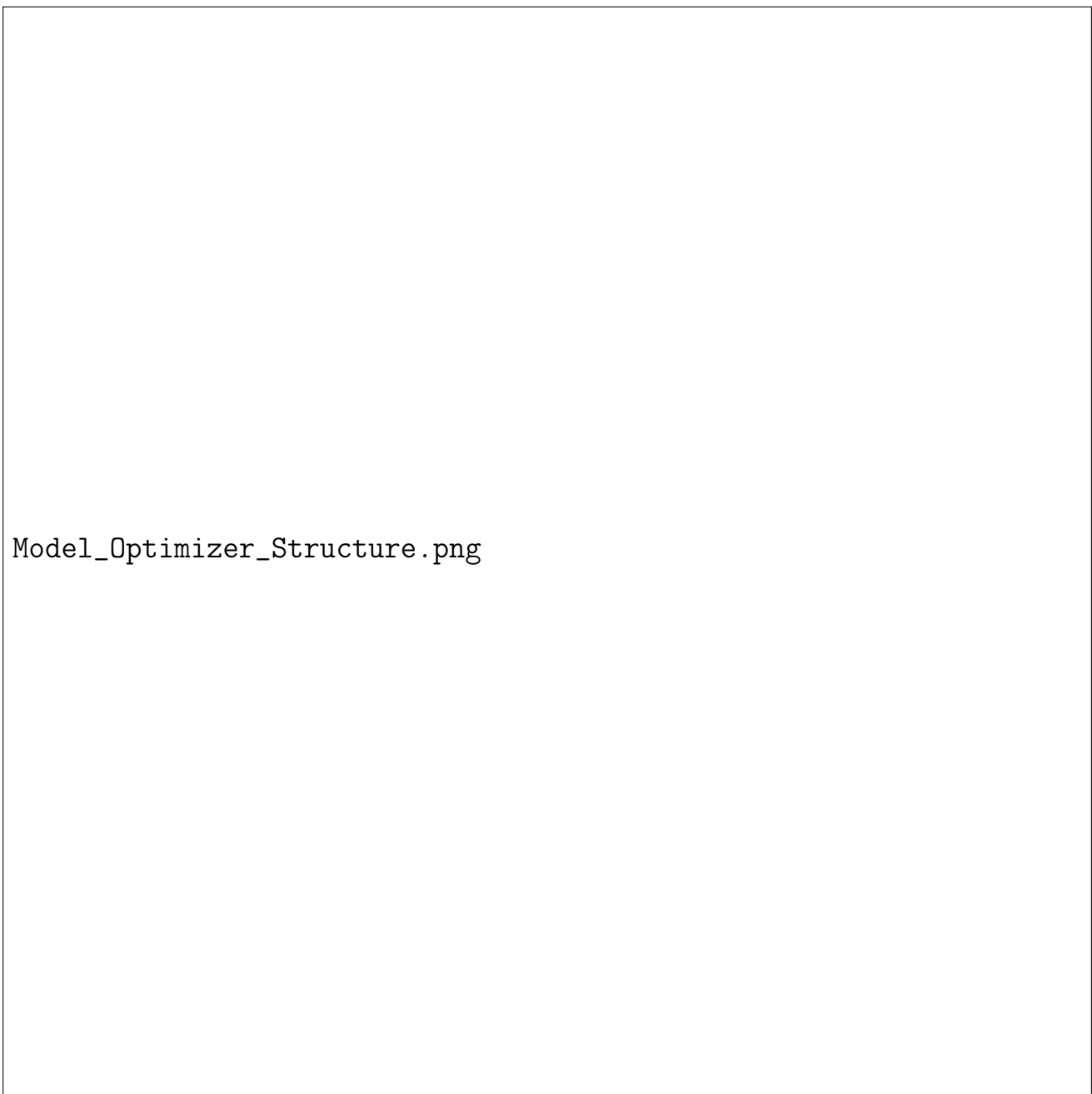
- [12] Model Optimizer Developer Guide. — Intel. https://docs.openvinotoolkit.org/latest/_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html
- [13] Deep Learning For Computer Vision. — Intel. <https://software.intel.com/en-us/openvino-toolkit/deep-learning-cv>
- [14] Обзор OpenVINO/ Open Model Zoo. — Intel. <https://delta-course.org/Intel-CV-School>
- [15] face-detection-adas-0001. — Intel. https://docs.openvinotoolkit.org/latest/_face_detection_adas_0001_description_face_detection_adas_0001.html
- [16] age-gender-recognition-retail-0013. — Intel. https://docs.openvinotoolkit.org/latest/_age_gender_recognition_retail_0013_description_age_gender_recognition_retail_0013.html
- [17] head-pose-estimation-adas-0001. — Intel. https://docs.openvinotoolkit.org/latest/_head_pose_estimation_adas_0001_description_head_pose_estimation_adas_0001.html
- [18] emotion-recognition-retail-0003. — Intel. https://docs.openvinotoolkit.org/latest/_emotions_recognition_retail_0003_description_emotions_recognition_retail_0003.html
- [19] cv::VideoCapture Class Reference. — OpenCV. https://docs.opencv.org/3.1.0/d8/dfe/classcv_1_1VideoCapture.html
- [20] Integrate the Inference Engine API with Your Application (Legacy). — Intel. https://docs.openvinotoolkit.org/latest/_docs_IE_DG_Integrate_with_customer_application.html
- [21] Процессор ELISE для систем компьютерного зрения. — Элвис. <https://multicore.ru/index.php?id=1406>
- [22] Buildroot. — Википедия. <https://en.wikipedia.org/wiki/Buildroot>
- [23] Чуприков Вячеслав. Разработка ПО для первичной обработки видеoinформации на основе системы «ЭЛИЗ-3D». — Кафедра Информационных систем и компьютерного моделирования. Волгоградский Государственный Университет. г. Волгоград, 2019 г.
- [24] Viola–Jones object detection framework. — Википедия. https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework

- [25] Метод Виолы — Джонса. — Википедия. https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%92%D0%B8%D0%BE%D0%BB%D1%8B_%E2%80%94%D0%94%D0%B6%D0%BE%D0%BD%D1%81%D0%B0
- [26] Признаки Хаара. — Википедия. https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%B7%D0%BD%D0%B0%D0%BA%D0%B8_%D0%A5%D0%B0%D0%B0%D1%80%D0%B0
- [27] OpenVX. — Википедия. <https://en.wikipedia.org/wiki/OpenVX>
- [28] Виктор Ерухимов. OpenVX: стандарт компьютерного зрения. — Хабр. <https://habr.com/ru/company/intel/blog/204236/>

The diagram area is mostly blank, suggesting the image content was not rendered or is obscured. The text 'OpenVINO_Structure.png' is visible on the left side of the diagram area.


OpenVINO_Structure.png

Рис. 10. Структура OpenVINO.



Model_Optimizer_Structure.png

Рис. 11. Типовой рабочий процесс для развертывания предварительно обученной модели



Inference_Engine.png

Рис. 12. Структура Механизма Вывода

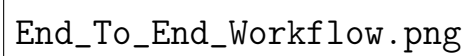
The image is a large, empty rectangular box with a thin black border. Inside the box, on the left side, is the text "End_To_End_Workflow.png" in a monospaced font.

Рис. 13. Рабочий процесс.

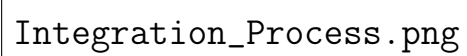
The image is a large, empty rectangular box with a thin black border. Inside the box, on the left side, is the text "Integration_Process.png" in a monospaced font.

Рис. 14. Общая схема работы приложения.