МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

НУКУССКИЙ ФИЛИАЛ ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ИМЕНИ МУХАММАДА АЛЬ-ХОРЕЗМИЙ

Кафедра «Информационные технологии»

направление Компьютерный инжиниринг(«ИТ-Сервис»)

Допускается к защите
Заведующий кафедрой
Туреniязова А.И.
_____
«____» _____ 2019 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему

«Разработка приложения для распознавания определенных объектов на платформе Android с использованием библиотеки OpenCV»

Выполнил: Казакбаев С.Б.

Научный руководитель: Баекеев Т.Р.

НУКУС - 2019 г.

ОГЛАВЛЕНИЕ

Введение

Поскольку смартфон представляет собой идеальное сочетание персонального цифрового помощника, медиаплеера, фотоаппарата и ряда других вещей – это полностью изменило прошлое о мобильном телефоне. В первые дни разработки приложений для смартфонов, развитие протекало только у мобильных компаний. После выпуска ОС Android в 2007 году разработка приложений для смартфонов стала пользоваться большим спросом. Android был разработан компанией Google с ядром Linux и с программным обеспечением GNU. Внедрение камеры в смартфона с обработкой видео в реальном времени становится очень популярным, в настоящее время это помогает решать важные вычислительные задачи. Практически во всех приложениях для смартфонов используется камера для использования технологии мобильного компьютерного зрения. Технологии мобильного компьютерного зрения играют жизненно важную роль в разработке приложений для повседневной деятельности. Эта технология имеет много целей, таких как поиск объектов, сегментирование, распознавание местоположения.

По мере развития процессоры смартфонов, таких компаний как MediaTek, ARM, NVIDIA Tegra и Snapdragon, расширяют возможности вычислений, демонстрируя быстрый рост мобильных приложений для компьютерного зрения, таких как редактирование изображений, дополненная реальность, распознавание объектов. Длительное время обработки из-за высокой вычислительной сложности препятствует использование алгоритмов мобильного компьютерного зрения в мобильных приложениях. Дабы преодолеть эту проблему, инженеры и разработчики исследовали такие библиотеки, как OpenGL и OpenCV. Разработчики приложений столкивались с множеством проблем, поскольку у них не было основной идеи для обработки видео в реальном времени. И решением этой проблемы стала библиотека

OpenCV, написанное на языке C, C ++, которая уменьшает сложность разработки и исследований.

Распознавание и обнаружение объектов в реальном времени является сложной и любимой областью исследований в современной быстрорастущей технологии мобильного компьютерного зрения. Компьютерному зрению нашлось применение в визуальном наблюдение, навигация робота и т.п. . Обнаружение и распознавание объектов в основном состоит из трех этапов: первый - извлечение признаков, вторая классификация, и третий - распознавание объекта с использованием машинного обучения и нескольких других технологий.

Глава I. Теоретические сведения о реализации приложения на платформе Android

Моя работа о распознавании определенного объекта с помощью библиотеки OpenCV на мобильным платформах Android. Эта глава будет охватывать две темы. Сначала я расскажу вам про основы Android. Что такое Android? Как это работает? Далее подробно проведу обзор существующих методов распознавания изображений и объектов на мобильных телефонах с последующим более подробным объяснением методов, которые я использовал в своем приложении. Во время изучения своей тему я исследовал аналогичные проекты распознавания образов от ведущих специалистов в этой, области. Присущи всему этому, можно сказать, что я приобрёл теоретическое знания в детектировании образов и развил свои навыки программирования.

1.1. Android

Android - это первая открытая и бесплатная платформа для мобильных устройств, разработанная членами Open Handset Alliance. Open Handset Alliance - это группа из более чем 40 компаний, включая Google, ASUS, Garmin, HTC, ... Эти компании собрались вместе, чтобы ускорить и улучшить разработку мобильных устройств. Тот факт, что это открытая платформа, имеет много преимуществ:

- Потребители могут покупать более инновационные мобильные устройства по более низким ценам.
- Операторы мобильной связи могут легко настраивать свои продуктовые линейки и быстрее предлагать новые услуги.
- Производители мобильных телефонов выиграют от более низких цен на программное обеспечение.

- Разработчики смогут легче создавать новое программное обеспечение, поскольку они имеют полный доступ к исходному коду системы и обширной документации API.

Райан Пол дает объяснение, касающееся лицензии на open source проекта Android. Как основной участник Android, Google решил выпустить что проект с открытым исходным кодом под версией 2 Apache Software License (ASL). ASL является разрешающей лицензией, что означает, что код, выпущенный по этой лицензией, может использоваться в продуктах с закрытым исходным кодом, в отличие от лицензий с авторским правом, таких как GPL. Это побуждает все больше компаний вкладывать средства в разработку новых приложений и сервисов для Android, поскольку они могут получить прибыль от этих улучшений для платформы. Обратите внимание, что ASL применяется только к компонентам пользовательского пространства платформы Android. Базовое ядро Linux лицензируется по GPLv2, а сторонние приложения могут быть выпущены под любой лицензией.

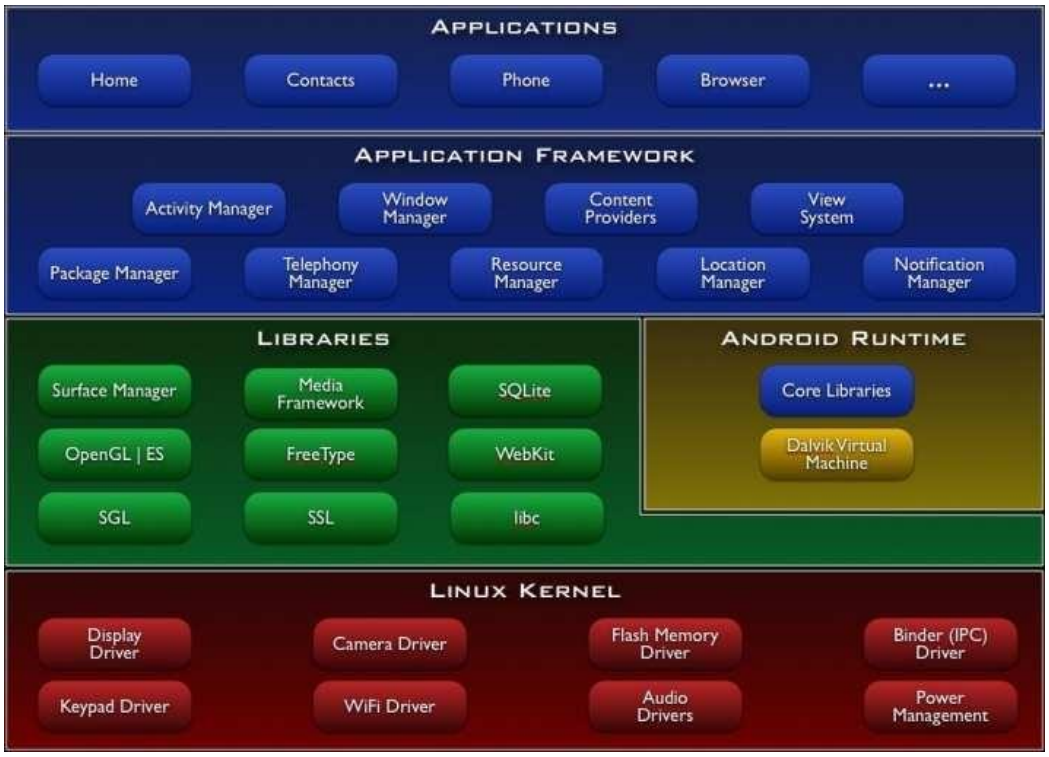


Рисунок 1.1: Архитектура системы Android

В следующих параграфах я объясню некоторые подробности о платформе Android, основываясь на информации, найденной на официальном сайте Android.

Среды разработки ANDROID STUDIO. В настоящее время трудно представить жизнь современного человека без мобильного телефона. Помимо своей основной функции телефоны играют значимую роль в каждодневной работе: при их помощи читают файлы, проверяют электронную почту, распечатывают различные документы с помощью сетевого принтера. В связи с этим на рынке появился самостоятельный сегмент – мобильные приложения. Особенность этого сегмента состоит в том, что создание приложений должно проводиться с учетом специфики мобильных устройств, а именно: различиями интерфейса, параметрами экрана, сенсорным управлением.

На сегодняшний день операционная система *Android* является наиболее распространенной в сфере мобильных устройств. Свою популярность Android получил во многом благодаря открытому исходному коду и политике Google, разрешающую производителям мобильных устройств безвозмездно использовать платформу Android для своей продукции. Android используется большинством крупных компании-производителей мобильных устройств, таких как Samsung, HTC, Sony, Huawei.

Android Studio – это интегрированная среда разработки (IDE) для работы с платформой Android, выпущенная компанией Google. Первая версия Android появилась 23 сентября 2008 года и получила название «Apple Pie». Изначально Google рассчитывала давать версиям Android имена известных роботов, но отказалась от этой идеи из-за проблем с авторскими правами. Каждая версия системы, начиная с версии 1.5, получает собственное кодовое имя на тему сладостей. Кодовые имена присваиваются в алфавитном порядке латинского алфавита. Платформа Android состоит из операционной системы, основанной на ядре операционной системы Linux, встроенных мобильных приложений и

переходного программного обеспечения. Во главе создания и совершенствования Android платформы стоит компания Google.

Android позволяет фоновое выполнение какого-либо действия, поддерживает двумерную и трехмерную графику, доступ к файловой системе и базе данных, обеспечивает обширную библиотеку элементов пользовательского интерфейса.

Архитектура системы Android состоит из следующих уровней:

1) Ядро операционной системы Linux.

2) Библиотеки и система выполнения.

3) Уровень каркаса приложений наделяет разработчика доступом к интерфейсу прикладного программирования API (application programming interface).

4) Уровень приложений – комплекс стандартных приложений.

На рисунке 1.2. представлена архитектура системы Android.

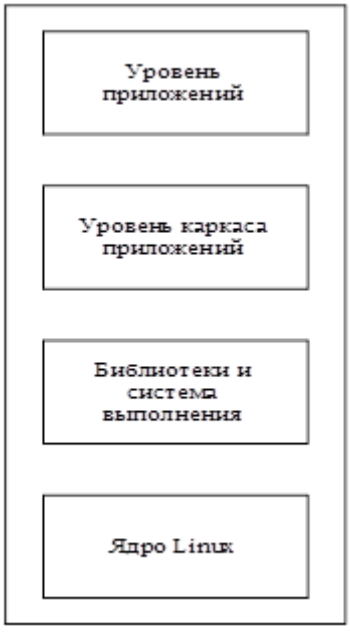


Рисунок 1.2: Изображение архитектуры системы Android

Разработчик, как правило, работает с уровнями приложений и каркаса приложений. Библиотеки, система выполнения и ядро операционной системы Linux от разработчика скрыты.

Одним из важнейших инструментов для разработки Android-приложений является универсальное средство разработки мобильных приложений для операционной системы Android (Android SDK) – комплекс средств программирования, содержащий инструменты, которые необходимы для создания, компиляции и сборки мобильного приложения.

В настоящее время создание программного обеспечения во многих случаях осуществляется с применением среды интегрированной разработки (IDE). В IDE автоматизирован процесс компиляции, сборки и запуска приложения, что облегчает работу программисту и позволяет начинающему разработчику без особых усилий впервые создать свое собственное приложение.

Существует две преимущественно популярные среды разработки, рекомендованные Google: Android IDE (ADT) и Android Studio.

Android IDE – среда разработки мобильных приложений под операционную систему Android, основанная на интегрированной среде разработки приложений Eclipse. Содержит встроенные инструменты для создания, компиляции, сборки и отладки мобильных приложений.

Android Studio – среда разработки мобильных приложений под операционную систему Android, основанная на интегрированной среде разработки программного обеспечения IntelliJ IDEA. Аналогично среде Android IDE, Android Studio содержит встроенные инструменты для создания и отладки мобильных приложений. Дополнительно ко всем возможностям в Android Studio реализованы:

- поддержка системы автоматической сборки Gradle;
- уникальная система рефакторинга кода;
- инструменты для поиска и устранения различных проблем;

- окно предварительного просмотра, которое показывает запущенное приложение сразу на нескольких устройствах и в реальном времени;

- поддержка облачной платформы Google Cloud Platform.

В данный момент компания Google прекращает поддержку инструментов для разработки в операционной системе Android для среды Android IDE.

В ходе изучения на практике среды разработки мобильных приложений Android Studio и написания небольших программ (приложение «Заметки» для создания пометок, приложение для определения долготы и широты объекта, приложение, определяющее местоположение с помощью Google Maps, приложение «Фонарик») были определены следующие недостатки Android Studio:

1) Необходимо иметь базовый уровень программирования на языке Java и знания английского языка.

2) Вследствие постоянного сопровождения разработчика системой автоматической сборки (Gradle) значительно замедляется время компиляции проектов.

3) Высокие системные требования для компьютера разработчика.

4) Стандартные эмуляторы требовательны по отношению к системным ресурсам, долго включаются и не обладают всеми возможностями реальных смартфонов.

5) Отсутствие русскоязычной справки.

Достоинства Android Studio:

1) Удобный дизайн.

2) Удобный конструктор интерфейсов, позволяющий просматривать отображение экрана на любом устройстве, вплоть до телевизоров и часов. Элементы интерфейса отображаются прямо так, как они будут выглядеть на определенной версии операционной системы.

3) Встроенный комплекс средств разработки SDK (software development kit), выдает уведомление с установкой необходимого API для запуска старого проекта.

4) Удобная структура проекта.

5) Наличие логов для отслеживания ошибок, процессов и потоков.

6) Наличие достаточно большого количества литературы на русском языке.

Обзор архитектуры. Android разработан, чтобы быть полным стеком операционной системы, Middleware и Application. На рисунке 1.1. представлен обзор его компонентов.

Android использует ядро Linux 2.6. Google решила использовать Linux в качестве уровня аппаратной абстракции, потому что у него есть рабочая модель драйвера и часто встречающие драйверы. Linux также обеспечивает управление памятью, управление процессами, безопасность, сети, и т.п.

Следующий уровень содержит библиотеки. Они написаны на C / C ++ и обеспечивают основную мощь платформы Android. Этот уровень состоит из нескольких различных компонентов. Сначала есть Surface Manager. Этот менеджер используется для отображения разных окон в нужное время. OpenGL | ES и SGL – графические библиотеки системы. Первая – это 3D-библиотека, вторая – 2D-библиотека. Их можно комбинировать для использования 2D и 3D в одном приложении. Следующий компонент – Media Framework. Эта структура предоставляется участником Open Handset Alliance, Packet Video. Он содержит большинство кодов, необходимых для обработки наиболее популярных форматов мультимедиа. Например: MP3, MPEG4, AAC, и многое подобного. FreeType отображает все растровые и векторные шрифты, представленные на платформе Android. Для хранения данных используется система управления базами данных SQLite. Webkit – это движок браузера с открытым исходным кодом, который используется как ядро браузера,

предоставляемого Android. Они оптимизировали его, дабы он хорошо отображался на небольших дисплеях.

На этом же уровне находится Android Runtime. Android Runtime содержит одну из самых важных частей Anroid: виртуальную машину Dalvik. Эта виртуальная машина использует Dex-файлы. Эти файлы являются байт-кодами, полученными в результате преобразования .jar-файлов во время сборки. Dex-файлы созданы для работы в ограниченных средах. Они идеально подходят для небольших процессоров, они эффективно используют память, их структуры данных, могут быть разделены между различными процессами, и они имеют высокую оптимизированность для процессора интерпретатор байт-кода. Кроме того, можно запустить несколько экземпляров виртуальной машины Dalvik. В среде выполнения Android мы также можем найти библиотеки ядра, содержащие все классы коллекций, утилиты, операции ввода-вывода и многого другого.

Прикладная платформа написана на Java и является инструментарием, который используют все приложения. Это приложения, поставляемые с телефоном, приложения, написанные Google, или приложения, написанные домашними пользователями. Диспетчер операций управляет жизненным циклом всех приложений. Диспетчер пакетов отслеживает приложения, установленные на устройстве. Window Manager – это абстракция JPL поверх сервисов более низкого уровня, предоставляемых Surface Manager. Диспетчер телефонии содержит API-интерфейсы для телефонных приложений. Поставщики контента образуют структуру, позволяющую приложениям обмениваться данными с другими приложениями. Например, приложение «Контакты» может делиться всей своей информацией с другими приложениями. Диспетчер ресурсов используется для хранения локализованных строк, растровых изображений и других частей приложения, которые не являются кодом. Система просмотра содержит кнопки и списки. Он также обрабатывает отправку событий, макет, рисование и тому подобное.

Диспетчер местоположения, Диспетчер уведомлений и Диспетчер XMPP также являются интересными API, которые мы опишем в следующих разделах.

Уровень приложений содержит все приложения: Главная страница, Контакты, Браузер, Пользовательские приложения и так далее. Все они используют Application Framework.

API's. В предыдущем перечислены некоторые API, содержащиеся в Application Framework. Теперь мы подробнее рассмотрим более важные из них:

- Менеджер местоположения
- XMPP Сервис
- менеджер уведомлений
- Просмотры

Менеджер местоположений позволяет получать географическую информацию. Он может дать вам знать ваше текущее местоположение и с помощью намерений может вызывать уведомления, когда вы приближаетесь к определенным местам. Этими местами могут быть заранее определенные местоположения, местоположение друга и так далее. Менеджер местоположений определяет ваше местоположение на основе доступных ресурсов. Если у вашего мобильного телефона есть GPS, Менеджер местоположений может определить ваше местоположение очень подробно в любой момент. В противном случае ваше местоположение будет определяться с помощью идентификаторов вышек сотовой связи или возможной географической информации, содержащейся в сетях WiFi.

Служба XMPP используется для отправки сообщений с данными любому пользователю под управлением Android на своем мобильном телефоне. Эти сообщения данных являются Интентами с парами значений. Эти данные могут быть отправлены любым приложением, поэтому они могут содержать информацию любого типа. Его можно использовать для разработки многопользовательских игр, программ чата и мессенджеров. Но также его можно использовать для отправки географической информации другим

пользователям. Например, вы можете отправить свое текущее местоположение своим друзьям. Таким образом, их менеджер местоположения может уведомить ваших друзей, когда вы находитесь в их районе. Служба XMPP работает с любой учетной записью Gmail, поэтому приложения могут использовать ее без создания какой-либо серверной инфраструктуры.

Диспетчер уведомлений позволяет добавлять уведомления в строку состояния. Уведомления добавляются из-за связанных событий (Intents), таких как новые SMS-сообщения, новые сообщения голосовой почты и так далее. Мощность уведомлений в строке состояния может быть распространена на любое приложение. Это позволяет разработчикам уведомлять пользователей о важных событиях, которые происходят в их приложении. Это много типов уведомлений, в то время как каждый тип представлен определенной иконкой в строке состояния. Нажав на значок, можно отобразить соответствующий текст. Если вы щелкните по нему еще раз, вы попадете в приложение, которое создало уведомление.

Представления используются для создания приложений из стандартных компонентов. Android предоставляет списки, сетку, галерею, кнопки, флажки и многое другое. Система просмотра также поддерживает несколько методов ввода, размеры экрана и раскладки клавиатуры. Кроме того, есть еще два инновационных взгляда. Первый - это MapView, представляющий собой реализацию механизма рендеринга карт, используемого в приложении «Карты». Это представление позволяет разработчикам встраивать карты в свои приложения, чтобы легко предоставлять географическую информацию. Представление может полностью контролироваться приложением: настройка местоположения, масштабирование, ... Второе инновационное представление - это веб-представление. Это позволяет встраивать html-контент в ваше приложение.

Структура приложения. При написании приложения для Android необходимо учитывать, что приложение можно разделить на четыре компонента:

1. Деятельность
2. IntentReceiver
3. Обслуживание
4. Поставщик услуг

Activity - это пользовательский интерфейс. Например, приложение Почта имеет 3 действия: список почты, окно для чтения писем и окно для составления писем. IntentReceiver - это зарегистрированный код, который запускается при запуске внешним событием. С помощью XML можно определить, какой код должен выполняться при каких условиях (доступ к сети, в определенное время, ...). Сервис - это задача, которая не имеет пользовательского интерфейса. Это долгоживущая задача, которая работает в фоновом режиме. ContentProvider - это часть, которая позволяет обмениваться данными с другими процессами и приложениями. Приложения могут хранить данные любым удобным для них способом, но чтобы сделать их доступными для других приложений, они используют ContentProvider. На рисунке 1.3. показано это использование ContentProvider.

Основным файлом в приложении Android является файл AndroidManifest.xml. Он содержит информацию, которая должна быть у системы, прежде чем она сможет запустить код приложения. Он описывает компоненты, упомянутые выше, например, он определяет, какое действие будет запущено при запуске приложения. Он также объявляет различные разрешения, такие как доступ в Интернет. Имя приложения и значок, используемые в приложении Home, также определены в этом файле. Когда понадобятся определенные библиотеки, это будет упомянуто здесь. Это лишь некоторые из многих вещей, которые делают файлы AndroidManifest.xml.

Android разработан для поощрения повторного использования и замены компонентов. Предположим, мы хотим написать приложение, которое хочет

выбрать фотографию. Вместо того, чтобы писать собственный код для выбора этой фотографии, мы можем использовать встроенный сервис. Для этого нам нужно сделать запрос или намерение. Затем система с помощью диспетчера пакетов определит, какое установленное приложение лучше всего соответствует нашим потребностям. В этом случае это будет встроенная фотогалерея. Если в дальнейшем мы решим заменить встроенную фотогалерею, наше приложение продолжит работать, потому что система автоматически выберет это новое приложение для сбора фотографий.

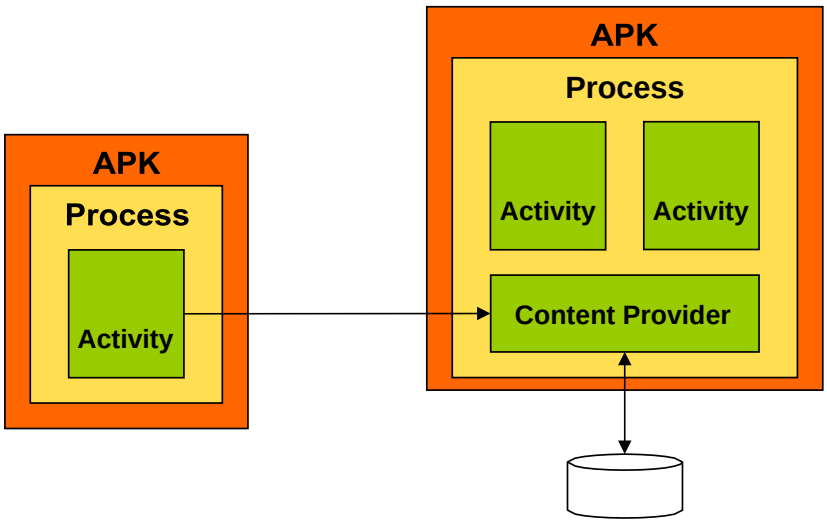


Рисунок 1.3: Обзор структуры базового приложения Android

Модель приложения: приложения, задачи, потоки и процессы. В приложении для Android есть три важных условия:

1. Пакет Android
2. Задача
3. Процесс

В то время как в других операционных системах существует сильная корреляция между исполняемым изображением, процессом, значком и приложением, в Android более плавная согласованность.

Пакет Android представляет собой файл, содержащий исходный код и ресурсы приложения. Так что это в основном форма, в которой приложения распространяются среди пользователей. После установки приложения на

экране появится новый значок для доступа к нему. На рисунке 1.4. показано, как создаются пакеты Android.

При нажатии на значок, созданный при установке пакета, появится новое окно. Пользователь увидит это как приложение, но на самом деле это задача, с которой он имеет дело. Пакет Android содержит несколько действий, одним из которых является точка входа верхнего уровня. При нажатии на значок, задача будет создана для этого действия. Каждое новое начатое действие будет выполняться как часть этой задачи. Таким образом, задача может рассматриваться как стек действий. Новая задача может быть создана с намерением действия, которое определяет, какая это задача. Это намерение можно запустить с помощью флага Intent.FLAG_ACTIVITY_NEW_TASK. Действия, запущенные без этого флага, будут добавлены к той же задаче, что и операция, запускающая ее. Если используется флаг, сходство задач, уникальное статическое имя для задачи (по умолчанию это имя пакета Android), будет использоваться для определения, существует ли уже задача с таким же сходством. Если нет, новая задача будет создана. В противном случае существующее задание будет перенесено, и действие будет добавлено в его стек действий.

Процесс - это то, о чем пользователь не знает. Обычно весь исходный код в пакете Android выполняется в одном низкоуровневом процессе ядра, как также показано на рисунке 1.3. Это поведение можно изменить, используя тег процесса. Основное использование процессов:

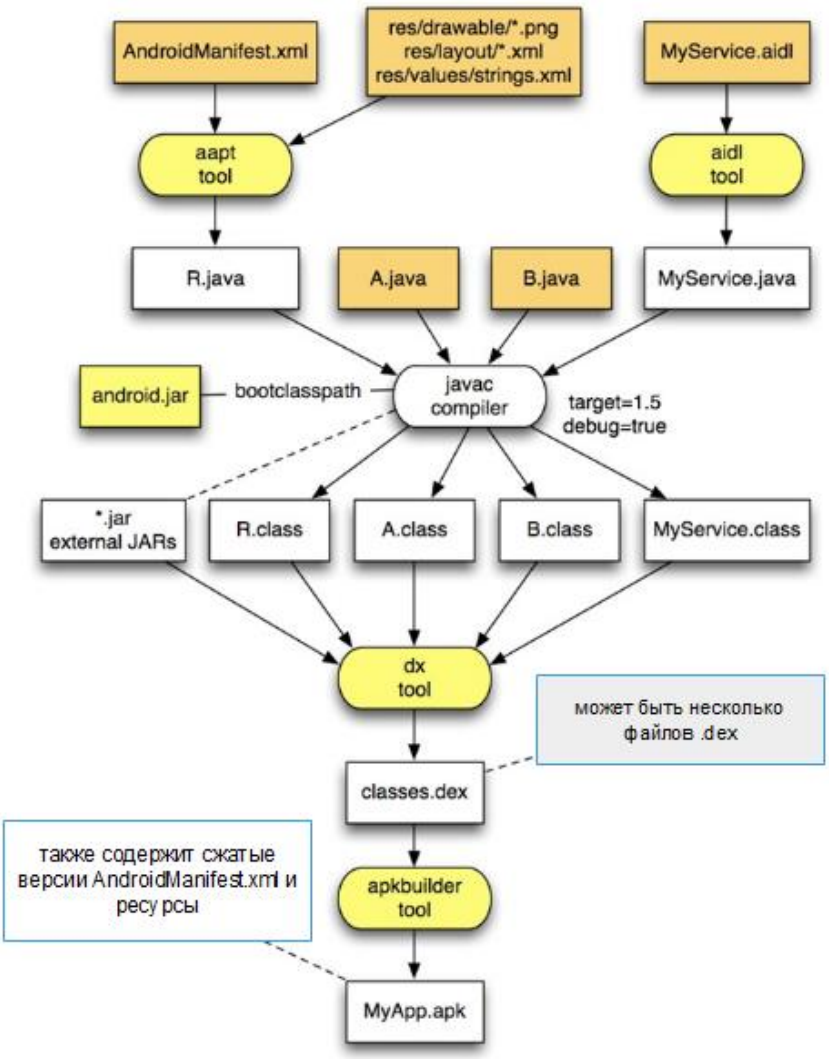


Рисунок 1.4: Процесс разработки пакета Android

- Повышение стабильности и безопасности за счет помещения ненадежного или нестабильного кода в отдельные процессы.
- Сокращение накладных расходов за счет запуска кода нескольких пакетов Android в одном процессе.
- Разделение тяжелого кода в процессах, которые могут быть уничтожены в любое время, без необходимости уничтожения всего приложения.

Обратите внимание, что два пакета Android, работающих с разными идентификаторами пользователей, не могут быть запущены в одном и том же процессе во избежание нарушения безопасности системы. Каждый из этих

пакетов получит свой отдельный процесс. Android пытается сохранить все свои запущенные процессы однопоточными, если приложение не создает свои собственные дополнительные потоки. Поэтому при разработке приложений важно помнить, что для каждого экземпляра Activity, BroadcastReceiver, Service или ContentProvider не создаются новые потоки. Таким образом, они не должны содержать длинные или блокирующие операции, поскольку они будут блокировать все другие операции в процессе.

Жизненный цикл приложения. В этом параграфе я собираюсь объяснить, как Android работает с разными приложениями, работающими одновременно. Сложность заключается в том, что на Android у нас ограниченные ресурсы. Поэтому управление памятью является одной из важнейших функций Android-системы. Конечно, фреймворк заботится об этом, и разработчик приложений или конечный пользователь даже не должен знать об этих событиях, происходящих в фоновом режиме. Чтобы обеспечить цельный и удобный интерфейс, Android всегда предоставляет пользователю кнопку «назад», которая работает во всех приложениях. Что бы ни делал пользователь, когда он нажимает кнопку «назад», должен появиться предыдущий экран.

Типичным примером для демонстрации и объяснения этой системы является пользователь, желающий прочитать свою электронную почту, щелкнув ссылку на веб-сайт, который открывается в его браузере. Пользователь запускается с домашнего экрана, запускаемого «домашним» системным процессом, где он щелкает, чтобы открыть почтовое приложение. Почтовое приложение запускается путем создания системного процесса для почтового приложения и по умолчанию запускает действие «список рассылки». Деятельность можно рассматривать как одно окно графического интерфейса. Там пользователь нажимает на электронную почту. Перед запуском этого нового окна для просмотра сообщения «системная посылка» сохраняется в системном процессе для почтовой рассылки, и эта посылка

содержит текущее состояние активности почтовой рассылки. Это необходимо, потому что в будущем системный процесс может быть удален из памяти (поскольку другому приложению нужна память), но когда пользователь возвращается к действиям со списком рассылки, он ожидает увидеть то же окно, которое видел раньше. Значение: прокрутка до той же страницы в списке, выбор правильного элемента. Каждый раз, когда начинается новое действие, либо из того же приложения, либо из другого, пакет состояний сохраняется в системном процессе старого действия. Возвращаясь к примеру, активность сообщения открывается, и пользователь может прочитать его сообщение электронной почты. Пользователь нажимает на URL веб-сайта внутри сообщения, поэтому нам нужно запустить процесс браузера. Кроме того, состояние активности сообщений сохраняется в процессе почтовой системы, и пользователь видит браузер, открывающий URL-адрес. Теперь предположим, что на странице есть ссылка на карту. Еще раз пользователь хочет посмотреть на это, он щелкает, и посылка с информацией о состоянии для браузера сохраняется в его системном процессе. Однако теперь у нас есть домашний экран, почтовое приложение и браузер, и нам не хватает памяти. Итак, мы должны убить приложение, чтобы освободить место. Главный экран не может быть убит: он нужен нам, чтобы система реагировала. Браузер также был бы плохим выбором для уничтожения, но почтовое приложение на пару позиций назад в стеке, и убивать безопасно. Приложение карт запускается, создает системный процесс и действие, и пользователь может перемещаться. Затем он хочет вернуться. Так что теперь активность браузеров восстановлена на первый план. Пакет состояний, который он сохранил при запуске приложения карт, можно отбросить, так как информация о состоянии браузера все еще загружается. Возвращаясь к бэк-стеку, есть активность сообщений из почтового приложения. Этот процесс больше не выполняется, поэтому мы должны запустить его снова. Чтобы сделать это, мы должны предоставить доступную память, и поскольку приложение карт больше не используется, оно уничтожается. Так что теперь почта запускается снова и открывает новую

активность сообщений. Это пустое действие без какого-либо полезного контента для пользователя. Однако в системном процессе мы сохранили информационную посылку, и теперь она используется для заполнения активности сообщения. Обратите внимание, что посылка не содержит содержимого самого действия (почтовое сообщение), а содержит только информацию, необходимую для перестроения исходного действия. В этом случае это может быть описание того, как получить почтовое сообщение из базы данных. Таким образом, почтовое приложение восстанавливает содержание активности сообщения. Пользователь видит сообщение электронной почты точно так же, как оно появилось в прошлый раз, не зная, что действие было прервано загрузкой приложения карт!

1.2. Язык программирования Java

Java — сильно типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). В настоящее время проект принадлежит OpenSource и распространяется по лицензии GPL. В OpenJDK вносят вклад крупные компании, такие как — Oracle, RedHat, IBM, Google, JetBrains. Так же на основе OpenJDK эти компании разрабатывают свои сборки JDK. Как утверждает компания Oracle — отличия между OpenJDK и OracleJDK практически отсутствуют за исключением лицензии, отрисовки шрифтов в Swing и некоторых библиотек, на которые лицензия GPL не распространяется. Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре с помощью виртуальной Java-машины. Дата официального выпуска — 23 мая 1995 года. На 2019 год Java — один из самых популярных языков программирования.

История создания. Изначально язык назывался Oak («Дуб»), разрабатывался Джеймсом Гослингом для программирования бытовых электронных устройств. Из-за того, что язык с таким названием уже существовал, вскоре Oak был переименован в Java. Назван в честь марки кофе Java, которая, в свою очередь, получила наименование одноимённого острова (Ява), поэтому на официальной эмблеме языка изображена чашка с горячим кофе. Существует и другая версия происхождения названия языка, связанная с аллюзией на кофе-машину как пример бытового устройства, для программирования которого изначально язык создавался. В соответствии с этимологией в русскоязычной литературе с конца двадцатого и до первых лет двадцать первого века название языка нередко переводилось как Ява, а не транскрибировалось.

В результате работы проекта мир увидел принципиально новое устройство, которое опередило своё время более чем на 10 лет. Устройство получило название Star7, но из-за большой стоимости в 50 долларов не смогло произвести переворот в мире технологии и постепенно забылось.

Star7 не пользовался популярностью в отличии от языка программирования Java и его окружения. Следующий этапом жизни языка стала разработка интерактивного телевидения. В 1994 году стало очевидным, что интерактивное телевидение было ошибкой.

В середине 1994 года широкого распространения набрал WWW (world web wide) где Java смог реализовать свои возможности при построении динамических веб страниц, точнее, вставок в них, которые получили название servlet. Данная технология имела огромный успех который и сейчас преследует язык.

Впоследствии язык стал использоваться для написания клиентских приложений и серверного программного обеспечения.

Основные особенности языка Java. Программы на Java транслируются в байт-код Java, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.

Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

Часто к недостаткам концепции виртуальной машины относят снижение производительности. Ряд усовершенствований несколько увеличил скорость выполнения программ на Java:

- применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы (JIT-технология) с возможностью сохранения версий класса в машинном коде,
- обширное использование платформенно-ориентированного кода (native-код) в стандартных библиотеках,
- аппаратные средства, обеспечивающие ускоренную обработку байт-кода (например, технология Jazelle, поддерживаемая некоторыми процессорами архитектуры ARM).

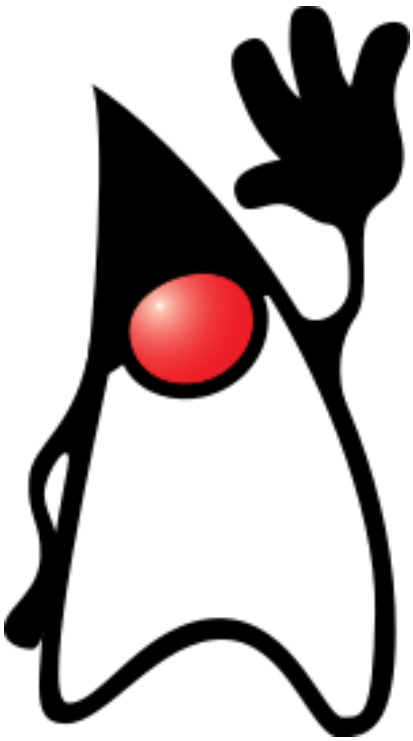


Рисунок 1.5: Дюк, талисман Java

По данным сайта *shootout.alioth.debian.org*, для семи разных задач время выполнения на Java составляет в среднем в полтора-два раза больше, чем для C/C++, в некоторых случаях Java быстрее, а в отдельных случаях в 7 раз медленнее. С другой стороны, для большинства из них потребление памяти Java-машиной было в 10—30 раз больше, чем программой на C/C++. Также примечательно исследование, проведённое компанией Google, согласно которому отмечается существенно более низкая производительность и бóльшее потребление памяти в тестовых примерах на Java в сравнении с аналогичными программами на C++.

Идеи, заложенные в концепцию и различные реализации среды виртуальной машины Java, вдохновили множество энтузиастов на расширение перечня языков, которые могли бы быть использованы для создания программ, исполняемых на виртуальной машине. Эти идеи нашли также выражение в спецификации общеязыковой инфраструктуры CLI, заложенной в основу платформы .NET компанией Microsoft.

1.3. OPENCV

OpenCV (англ. *Open Source Computer Vision Library*, библиотека компьютерного зрения с открытым исходным кодом) — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков.

Компьютерное зрение. Компьютерное зрение (иначе техническое зрение) — теория и технология создания машин, которые могут производить обнаружение, отслеживание и классификацию объектов.

Как научная дисциплина, компьютерное зрение относится к теории и технологии создания искусственных систем, которые получают информацию из изображений. Видеоданные могут быть представлены множеством форм, таких как видеопоследовательность, изображения с различных камер или трехмерными данными, например с устройства Kinect или медицинского сканера.

Как технологическая дисциплина, компьютерное зрение стремится применить теории и модели компьютерного зрения к созданию систем компьютерного зрения. Примерами применения таких систем могут быть:

1. Системы управления процессами (промышленные роботы, автономные транспортные средства).
2. Системы видеонаблюдения.
3. Системы организации информации (например, для индексации баз данных изображений).
4. Системы моделирования объектов или окружающей среды (анализ медицинских изображений, топографическое моделирование).
5. Системы взаимодействия (например, устройства ввода для системы человеко-машинного взаимодействия).

6. Системы дополненной реальности.
7. Вычислительная фотография, например для мобильных устройств с камерами.

Компьютерное зрение также может быть описано как дополнение (но не обязательно противоположность) биологическому зрению. В биологии изучается зрительное восприятие человека и различных животных, в результате чего создаются модели работы таких систем в терминах физиологических процессов. Компьютерное зрение, с другой стороны, изучает и описывает системы компьютерного зрения, которые выполнены аппаратно или программно. Междисциплинарный обмен между биологическим и компьютерным зрением оказался весьма продуктивным для обеих научных областей.

Обработка изображений — любая форма обработки информации, для которой входные данные представлены изображением, например, фотографиями или видеокадрами. Обработка изображений может осуществляться как для получения изображения на выходе (например, подготовка к полиграфическому тиражированию, к телетрансляции и т. д.), так и для получения другой информации (например, распознание текста, подсчёт числа и типа клеток в поле микроскопа и т. д.). Кроме статичных двухмерных изображений, обрабатывать требуется также изображения, изменяющиеся со временем, например, видео.

Большинство методов обработки одномерных сигналов (например, медианный фильтр) применимо и к двухмерным сигналам, которыми являются изображения. Некоторые из этих одномерных методов значительно усложняются с переходом к двухмерному сигналу. Обработка изображений вносит сюда несколько новых понятий, таких как связность и ротационная инвариантность, которые имеют смысл только для двухмерных сигналов. В обработке сигналов широко используются преобразование Фурье, а также вейвлет-

преобразование и фильтр Габора. Обработку изображений разделяют на обработку в пространственной области (преобразование яркости, гамма коррекция и т. д.) и частотной (преобразование Фурье, и т. д.). Преобразование Фурье дискретной функции (изображения) пространственных координат является периодическим по пространственным частотам с периодом 2pi.

Глава II. Основные методы обработки изображений для мобильных приложений

В этой главе будут описаны основные методы обработки изображений, которые необходимы для решения нашей проблемы. Эти методы включают калибровку камеры, обнаружение границ, переключение изображений и сопоставление шаблонов, которые являются ключами к созданию множества мобильных приложений для работы с изображениями в режиме реального времени.

2.1. Калибровка камеры на мобильных устройствах

Калибровка камеры в контексте компьютерного зрения — это процесс определения геометрических и оптических характеристик камеры (внутренние параметры камеры) и/или 3D положения и ориентацию кадра камеры относительно определенной системы координат. Это очень важный шаг для извлечения исходной информация из 2D-изображения, особенно в приложениях, направленных на решение количественного измерения, таких как глубина от стереоскопии и движения от изображений.

За последние десятилетия была проделана большая работа по калибровке камеры, мистер Чжан [Zhang, 2000] классифицировал эти методы примерно на две категории: фотограмметрическия калибровка и автокалибровка. Фотограмметрическая калибровка также рассматривается как трехмерная эталонная объектная калибровка. Калибровка камеры осуществляется за наблюдением объекта калибровки, геометрия которого в 3D-пространстве известна с очень хорошей точностью. Автокалибровка производится путем вычисления соответствий путем перемещения камеры в статической сцене, а затем восстанавливается как внутренние, так и внешние

параметры через принятие множеств изображении. Этот процесс может помочь восстановить 3D-структуры. Однако, есть много параметров для оценки с помощью автокалибровки, которая может привести к получению недостоверных результатов. Таким образом, в этом работе, фотограмметрическая калибровка будет использована.

Калибровка и переназначение (переназначение-это процесс извлечения пикселей из одного место на изображении и размещение их в другом положении на новом изображении) может исправить искажение длины и определение отношения между естественными единицами камеры (пикселями) и единицы реального мира. Проективное преобразование отображает точки в физическом мире с координатами (X ,Y,Z) к точкам на плане проекции с координатами (X,Y), с этой простой формой:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad \text{Уравнение 2.1}$$

Здесь w добавляется к (x, y) однородным координатам, связанным с точкой в проективном пространстве. Тогда $w = Z$ может быть вычислено. Параметры f_x и f_y являются фокусным расстоянием камеры, а c_x и c_y - оптические центры, выраженные в пиксельных координатах. Если для обеих осей, общее фокусное расстояние используется с заданным соотношением сторон α (обычно 1), затем $f_y = f_x \times \alpha$, а в верхней формуле есть одно фокусное расстояние. Матрица содержащий эти четыре параметра называется матрицей камеры.

Искажение камеры - это нелинейная модификация изображений, вызванная оптическими линзами. Искажение камеры в основном вызвано радиальными и тангенциальными факторами. Наличие радиальnouj искажение проявляется в виде эффекта «бочка» или «рыбий глаз». На практике радиальные искажения невелики и могут быть охарактеризованы с помощью

первых нескольких членов ряда Тейлора расширение около $r = 0$. Таким образом, будет использоваться следующая формула:

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \qquad \text{Уравнение 2.2}$$

$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \qquad \text{Уравнение 2.3}$$

где (x, y) - исходное местоположение (на тепловизоре) искаженной точки и $(x_{corrected}, y_{corrected})$ - новое местоположение в результате коррекции.

Тангенциальные искажения обусловлены производственными дефектами, вызванными тем, что объектив не точен в параллельной плоскости изображения. Тангенциальное искажение минимально и характеризуется двумя дополнительными параметрами, $p1$ и $p2$, следующим образом:

$$x_{corrected} = x + (2p_1 y + p_2(r^2 + 2x^2)) \qquad \text{Уравнение 2.4}$$

$$y_{corrected} = y + (p_1(r^2 + 2y^2) + 2p_2 x) \qquad \text{Уравнение 2.5}$$

Таким образом, всего существует пять параметров искажения, которые объединены в один вектор искажения:

$$Distraction_{parametrs} = (k_1, k_2, p_1, p_2, k_3) \qquad \text{Уравнение 2.6}$$

В этой работе OpenCV используется для проведения необходимой калибровки и вычисления внутренних параметров в уравнении(табл.х.6). Процедура калибровки направлена на камеру уже известной структуры (например, шахматная доска), которая содержит много отдельных и идентифицируемых точек. Взяв несколько ракурсов под разными углами, можно вычислить местоположение каждого изображения и ориентацию камеры. Кроме того, свойственные параметры камеры могут быть рассчитаны.

В этой работе шахматная доска используется в качестве известной структуры. Около пятнадцати изображений взятых из нескольких ракурсов, каждый шаблон которого равен новому уравнению. Использование этих изображений в качестве входа функции калибровки, выхода камеры во внутреннею матрицу, коэффициентов искажения, векторов вращения и

векторов перемещения будут рассчитываться. Необходима коррекция искаженных изображений с параметрами. процесс искажения состоит в том, чтобы вычислить карту искажения с помощью встроенной матрицы камеры и коэффициентов искажения, и применить эту карту к исходным изображениям.

Этапы калибровки можно перечислить следующим образом:

1. Программа загрузит параметры калибровки, такие как, где получить входное изображение;
2. Получите входные изображения шахматной доски с камеры в реальном времени или с сохраненные изображения;
3. Найти шаблоны из входных изображений и положение угла каждого шаблона на шахматной доске
4. Калибровка будет применяться с такими параметрами, как набор угловых точек и размер входного изображения, и рассчитать матрицу коэффициента искажения и матрица камеры.
5. В результате последнего шага для входных изображений выполняется искажение.
6. Сохраните результат калибровки и покажите неискаженные изображения

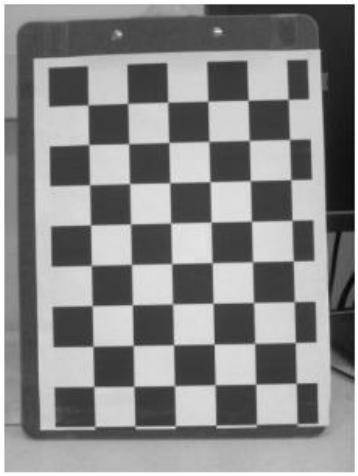
Таблица 3.1 показывает, как реализовать эти шаги в псевдокодах с использованием функций OpenCV. Поскольку процесс калибровки для определенной камеры нужно сделать хотя бы один раз, данные калибровки сохраненные в коде, могут быть применены к любой обработке изображений, основанной на этой камере. Таким образом, одно мобильное устройство должно пройти этот процесс только один раз, и сохранить данные калибровки для будущего использования. Все изображения, которые будут обработаны позже в этой работе из откалиброванных камер.

Таблица 2.1. Псевдокод калибровки камеры с использованием функций OpenCV

```
function camera_calibration
{
Read settings file;
if failed
print “Invalid setting”;
return false;
else
return true;
for (i = 0; i++; i <= end of the input image list)
{
ImageSet [i] = Read input image [i];
CC [i] = run findChessboardCorners();
if (i = end || false)
break;
}
Camera calibration:
NewImageSet = undistortion(ImageSet, calibrationData);
Show NewImageSet;
}
```

На рисунке 2.1. показаны результаты калибровки на шахматной доске. Обратите внимание, что в зависимости от аппаратное обеспечение камеры, результат калибровки может быть очевидным или наоборот. Здесь небольшая разница между оригинальным и неискаженным изображением, что означает, что эта камера имеет высокое качество с небольшим искажением. Сильные искажения могут возникнуть на нижнем конце и процессора камеры.

(a)



(b)

(c)

Рисунок 2.1. Калибровочная шахматная доска **(a)** набор оригинальных изображений шахматной доски, сделанных с разных ракурсов и расстояний; **(b)** исходное изображение; **(b)** откалиброванное изображение, примечание по сравнению с **(c)**, его край не искажён (как показано стрелкой).

2.2. Обнаружение границ на изображении

Границы являются характерными особенностями изображения, которые полезны для анализа изображения и классификации в широком спектре приложений. Исследование обнаружения границ существует уже долгое время и также существует множество алгоритмов. Обнаружение границы является важным шагом для многих других высокоуровневых обработок изображений и сложных изображений. Среди многих методов обнаружения границ один из самых используемых алгоритмов является детектор границ Канни.

Детектор границ Canny был разработан Дж. Ф. Кэнни в 1986 году, также известный как оптимальный детектор. Оригинальный алгоритм имеет следующие преимущества:

1) Низкий уровень ошибок: хорошее обнаружение только существующих ребер.

2) Хорошая локализация: расстояние между граничными пикселями и реальной границей в пикселях должно быть минимизированым.

3) Минимальный отклик: только один отклик детектора на ребро.

OpenCV использует детектор границ Canny для поиска границ на изображении. Для реализации в алгоритме первым шагом является сглаживание шумов изображения с использованием гауссовского фильтра. Гауссовский фильтр определяется размером ядра. Одним из негативных последствий является то, что этот фильтр будет слегка размытым от оригинального изображения. Например, ядро Гаусса размером равным 5, часто используется как показано в уравнении 3.7.

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \qquad \text{Уравнение 2.7}$$

Алгоритм Канни использует четыре фильтра для обнаружения горизонтальных, вертикальных и диагональных ребер в сглаженном изображении. Он применяет пару масок свертки в обоих направлениях x и y. Оператор обнаружения ребер возвращает значение для первой производной в направлениях. Эти две маски в x и y — G_x и G_y соответственно,

.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad \text{Уравнение 2.8}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \qquad \text{Уравнение 2.9}$$

Сила и направление градиента могут быть вычислены с помощью:

$$G = \sqrt{G_x^2 + G_y^2} \qquad \text{Уравнение 2.10}$$

$$\theta = \tan^{-1} \frac{G_X}{G_Y} \qquad \text{Уравнение 2.11}$$

Угол направления кромки округляется до одного из четырех углов, представляющих собой вертикаль, горизонталь и две диагонали в OpenCV.

Следующий шаг называется не-максимальное подавление. Он ищет изображение с оценки градиентов изображения, чтобы определить, предполагает ли величина градиента локальный максимум в направлении градиента. Это удаляет пиксели, которые не считаются частью границы и останутся только тонкие линии.

Алгоритм Canny пытается собрать отдельные пиксели-кандидаты границы в связанные контуры. Эти контуры сформированы путем прикладывания порог гистерезиса к пикселям. В алгоритме Canny есть два

порога, верхний и нижний. Если пиксель имеет градиент больше чем верхний порог, тогда принять его как пиксель границы; если пиксель под нижним порогом, он отклоняется. Если градиент пикселя находится между порогами, то он будет принимается только в том случае, если он подключен к пикселю, превышающему высокий порог. Canny рекомендуется соотношение высокого к низкому порогам между 2 к 1 и 3 к 1.

В библиотеке OpenCV все алгоритмы обработки изображений упакованы в Imgproc.class. Этот класс содержит различные методы обработки изображений, в том числе фильтрация изображений, геометрические преобразования изображений, различные преобразования изображений, гистограммы, структурный анализ и дескрипторы формы и обнаружение признаков. Обнаружение границ Canny относится к фильтрации изображений. Он упакован в одну функцию под названием Canny с пятью параметрами, включая входное изображение, выходное изображение, два порога и размер диафрагмы для операторов направления (уравнение 3.8 / 3.9).

При внедрении этого детектора границы в приложение Android, некоторые из шагов упомянутые в предыдущем описании изменены. Первый шаг – преобразование входного изображения. В шкале серого, что ускорит вычисления. Тогда фильтр Гаусса будет применяется к изображению, так как необходимо управление параметром сигма в гауссовском фильтре, чтобы избавиться от некоторых шумных границ. С необходимыми параметрами в оригинальной функции Canny, результат обнаружения границы контролируется путем изменения этих параметров.

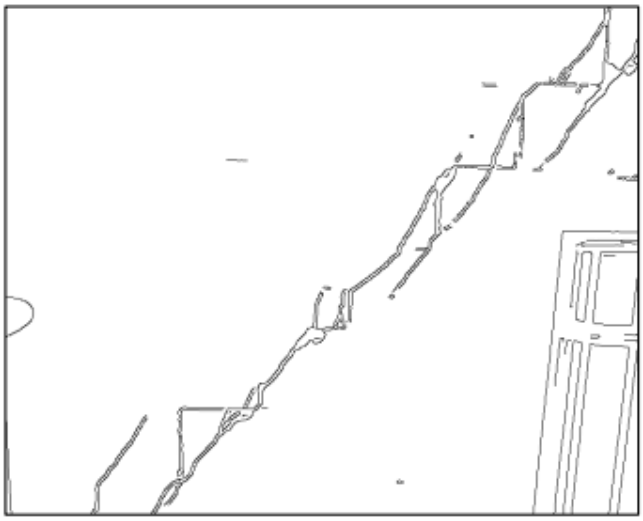
Псевдокод для реализации базового обнаружения границ Канни, показан в таблице 2.2. Замечено, что если пороги и сигма в коде установлены как динамические параметры, визуализация на основе взаимодействия в реальном времени и обнаружение границы могут быть реализованы. Поскольку мобильное устройство может отображать полученное изображение на экране, пользователь мобильного устройства может изменять эти параметры в режиме реального времени с помощью ScrollView.

Таблица 2.2. Псевдокод обнаружения границы с использованием функций в OpenCV

```
Function Edge_detection
{
if (switch = static)
inputImage = Read image from file;
else
inputImage = Read frames from camera in real
time;
Color converting;
BlurredImage = Run GaussianBlur (inputImage, Sigma);
EdgeImage = Run Canny(BlurredImage,Parameters);
Show EdgeImage;
}
```

Изображение трещины на стене и результат обнаружения границы показаны на рисунке 2.2. Параметры были настроены для получения наилучшего визуального результата, это означает только трещину, интерес пользователя остается на изображении, вся другая шумная информация отфильтровывается.

(a)



(b)

Рисунок 2.2. Обнаружение границы Canny: (a) исходное изображение трещины на стене (b) граница Canny — обнаружение применяется к изображению, с оптимальным значением параметра, только границы трещины как показано

2.3. Сшивание изображения при обработке изображений

Сшивание является фундаментальной задачей при обработке изображений, используемой для совмещения двух или более снимков, сделанные в различное время, с помощью камер или с разных углов обзора. В библиотеке OpenCV реализована автоматическая сшивка кадров. Сшивку кадров можно разделить на изображение регистрации и композитинг.

Регистрация влечет за собой этап обнаружения функции. Есть несколько функций алгоритмов обнаружения в OpenCV, такие функции обнаружение, ка функции Orb и функция SIFT. После того, как объекты были извлечены из всех *n* изображений, их можно сопоставить. Каждый объект сопоставляется с его *k* ближайшими соседями в пространстве объектов, используя дерево k-d для

нахождения примерно ближайших соседей. K-d дерево – это двоичное пространство, выровненное по оси, которое рекурсивно разделяет пространство признаков на среднее значение в измерении с самым высокой дисперсией. Затем он принимает функции всех изображений и проводит попарные совпадения между всеми изображениями и оцениваает повороты всех камер. Для того чтобы найти все подходящие изображения, OpenCV использует консенсус случайной выборки (RANSAC) выборка наборов вкладышей, которые совместимы с гомографией между изображениями и применяет вероятностную модель для проверки соответствия. После парных действий установленная между изображениями последовательность соответствующих изображений может быть найдена. В заключение, с помощью волновой коррекции и окончательной оценки панорамы получают зарегистрированный набор изображений.

В части композитинга OpenCV использует регистрационные данные, чтобы выровнить изображения. Предполагается, что горизонтальная ось камеры лежит в плоскости, поэтому плоскость, которая содержит центр камеры и горизонт могут быть вычислены. Применение глобального поворота удаляет волнистый эффект от выхода.

Для лучшего обзора результата сшивания, OpenCV также корректирует компенсацию ошибок экспозиции. Функция ошибки определяется для всех изображений. Функция является суммой компенсаций усиления нормализации ошибок интенсивности для всех перекрывающихся пикселей.

Последний шаг – это смешивание изображений. Есть два типа смешивания в OpenCV, *Feather Blender* и *Multi-band Blender*. Feather Blender – это метод линейного смешивания; оно может привести к ошибкам неправильной регистрации из-за неправильного моделирования камеры, радиальным искажениям и т.п. Multi-band Blender смешивает низкие частоты в течение большого пространственного диапазона и высокие частоты на небольшом расстоянии. Этот алгоритм может сохранить плавные переходы между изображениями и сохранять высокочастотные детали.

В этой работе, учитывая специфику мобильной визуализации, упрощенный конвейер реализован в рабочем процессе, как показано на Рисунке 2.3. При реализации этого конвейера, тест из набора изображений, показывающих длинные трещины повреждения на стене. Этот сценарий довольно част, когда полевые инженеры хотят записать повреждения. Они имеют тенденцию фотографировать много таких изображения по части трещин, а затем надеются сшить их вместе. Рисунок 2.4 показывает результат.

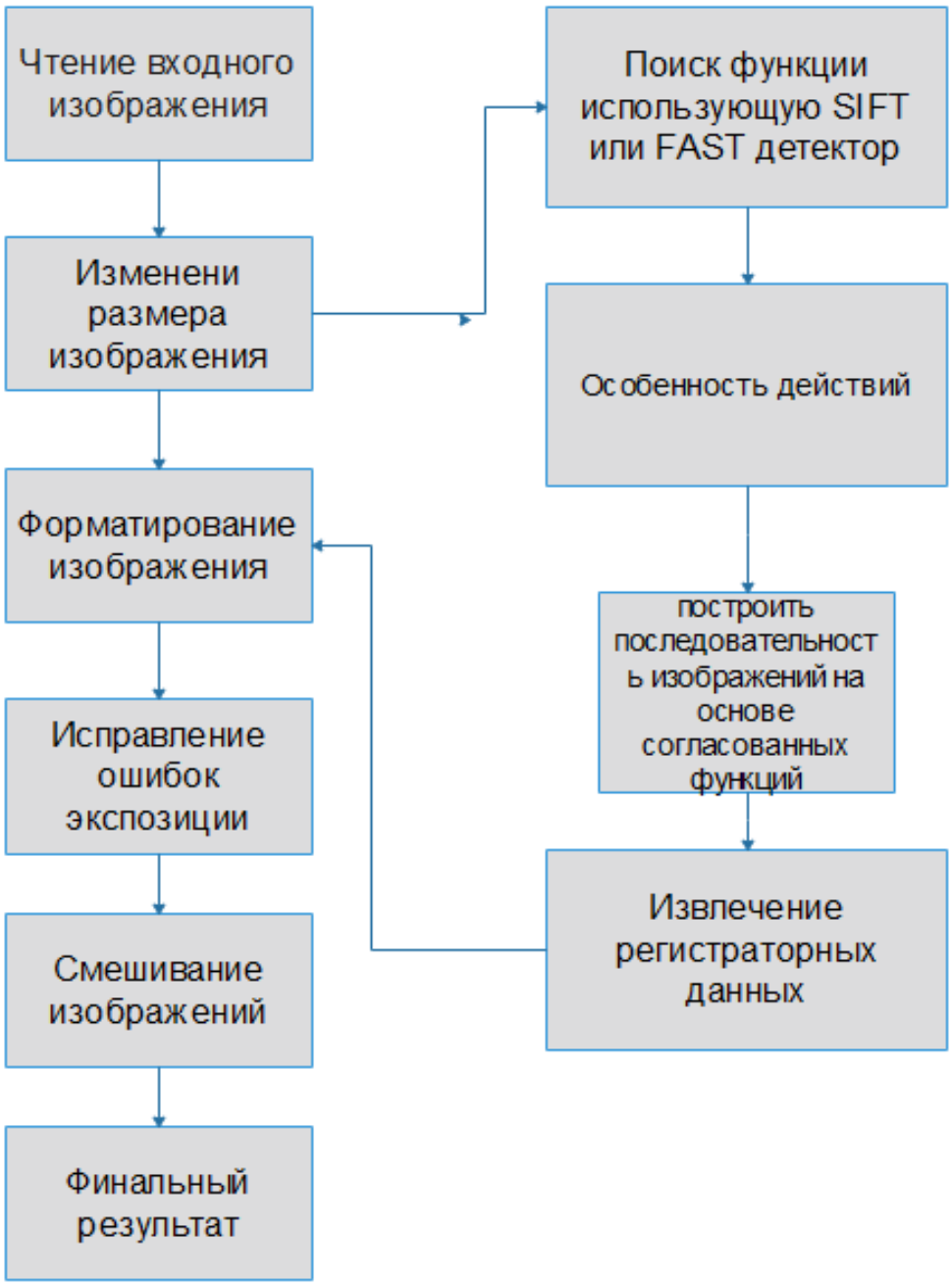


Рисунок 2.3. Конвейер сшивания изображений на платформе Android

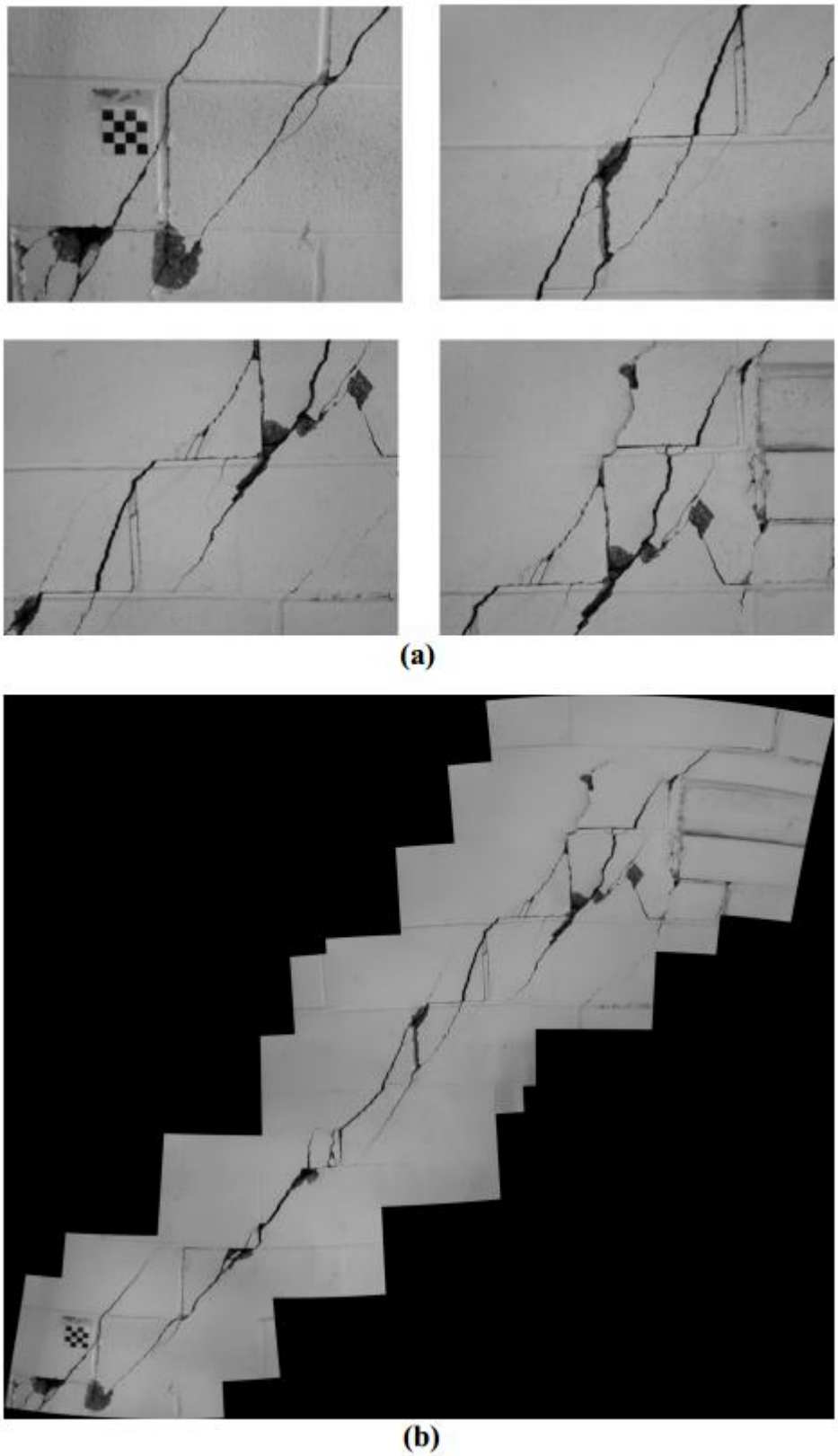


Рисунок 2.4. Иллюстратор сшивания: **(a)** локальные изображения трещины **(b)** сшитое изображение с 10 изображений, показывающих панорамный вид без потери детализации

2.4. Соответствие шаблона поиска

Сопоставление с шаблоном - это классический подход к проблеме поиска объекта в изображении. Изображение будет искать объект, применяя шаблон под каждым углом обзора на изображении. Шаблон обычно содержит стандартный объект интереса и является обычно меньше исходного изображения. Решение о наличии объекта в данном месте производится сравнением этого шаблона с заранее заданным порогом.

Типичный процесс сопоставления шаблона включает в себя взаимную корреляцию шаблона с изображенной сценой и вычислением меры сходства между ними, чтобы определить расстояние. Поскольку оценка корреляции является вычислительно дорогой, существует потребность в дешевых алгоритмах корреляции для обработки в реальном времени. Обширно было предложено несколько алгоритмов корреляционного типа мистером Vanderbrug and мистером Rosenfeld. Одним из методов является использование изображения пирамидой, как для шаблона, так и для изображения сцены, и выполнения регистрации поиска сверху вниз. Другие методы быстрого сопоставления используют двухпроходные алгоритмы; используйте подшаблон в грубо разнесенной сетке в первом проходе и нахождения лучшего соответствие в окрестности ранее найденных позиций во втором проходе.

В OpenCV - нисходящий алгоритм поиска в космической области, который также известен, как уровень пикселей, используется для завершения сопоставления. Изображение, в котором соответствует ожидание, тот шаблон, который нужно найти, называется исходным изображением I, в то время как сравнение с исходным изображением называется шаблоном изображения T. Для определения соответствия области, необходимо сравнить изображение шаблона с исходным изображением, сдвинув его. Это перемещает шаблон по одному пикселю за раз (слева направо или сверху вниз). На каждом

местоположение, метрика расстояния рассчитывается таким образом, чтобы показать, насколько шаблон похож на определенную область исходного изображения. И для каждого места шаблон изображения T более исходно изображению I, то оно будет хранить метрику в матрице результатов R. Каждое местоположение R содержит метрики совпадения, наибольшее значение R означает наилучшее возможное местоположение совпадения.

В OpenCV реализовано шесть методов сопоставления шаблонов.

Соответствующие отношения в терминах R и T описываются следующим образом:

- Методы сопоставления разности квадратов: **CV_TM_SQDIFF**

  Этот метод соответствует разнице в квадрате, поэтому идеальное совпадение будет 0 и плохие совпадения будут большой ценностью.

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2 \qquad \text{Уравнение 2.12}$$

- Методы сопоставления корреляций: **CV_TM_CCORR**

  Этот метод мультипликативно сопоставляет шаблон с изображением. Идеальное совпадение будет максимальным, а плохое совпадение будет минимальным или 0.

$$R(x,y) = \sum_{x',y'} (T(x',y') \times I(x+x',y+y'))^2 \qquad \text{Уравнение 2.13}$$

- Методы сопоставления коэффициентов корреляции: **CV_TM_CCOEFF**

  Этот метод сопоставляет шаблон относительно его среднего значения с изображением относительно его среднего. Следовательно, идеальное совпадение будет равно 1, а несовпадение будет равно -1; значение 0 просто означает отсутствие корреляции (случайное выравнивание).

$$R(x,y) = \sum_{x',y'} (T'(x',y') - I'(x+x',y+y'))^2$$

$$T'(x',y') = T'(x',y') - 1/(w \times h) \times \sum\nolimits_{x'',y''} T(x'',y'')$$

$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \times h) \times \sum\nolimits_{x'',y''} I(x+x',y+y')$$

Уравнение 2.14

Для трех методов, описанных выше, есть также нормализованные версии. Нормализованные методы могут помочь уменьшить последствия различия освещения между шаблоном и изображением. В каждом случае нормализации коэффициент тот же. Таким образом, эти нормализованные методы показаны ниже:

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \times \sum_{x',y'} I(x+x',y+y')^2}}$$

Уравнение 2.15

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \times I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \times \sum_{x',y'} I(x+x',y+y')^2}}$$

Уравнение 2.16

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \times I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \times \sum_{x',y'} I'(x+x',y+y')^2}}$$

Уравнение 2.17

Обычно существует компромисс между точностью и скоростью вычислений для вышеуказанных методов. Как правило, согласование разностей квадратов проще, но менее точно, тогда как сопоставление

коэффициентов корреляции является наиболее сложным и требует больших вычислительных мощностей.

Для реализации сопоставления с шаблоном необходимо предварительно сохранить изображение шаблона на диске или в памяти и прочитать кадр камеры в качестве исходного изображения. Затем проведите шаблон через исходное изображение. После того, как программа вычислит результат сопоставления, функция minMaxLoc будет использоваться для определения наивысшего или наименьшего (в зависимости от методов сопоставления) значения в матрице R, которое является наилучшим совпадением. Кроме того, должна быть обеспечена область хорошего соответствия вокруг этой наилучшей точки сопоставления, чтобы избежать случайного выравнивания шаблонов, которое просто хорошо работает. Так как в реальных проблемах соответствия, небольшие смещение шаблона не должно приводить к слишком сильному изменению результата, а это означает, что хорошая точка соответствия содержит рядом хорошие точки соответствия. Поэтому программы будут слегка сглаживать результирующее изображение перед использованием minMaxLoc для поиска максимального или минимального значения. Наконец, на исходном изображении отображается прямоугольник, чтобы показать результат пользователям приложения. Следующий псевдокод описывает эту процедуру.

Таблица 2.3. Шаблон соответствия псевдокода с использованием функций в OpenCV

```
function Template Matching
{
templateImage = Read image from file;
inputImage = Read frames from camera in real time;
method = Choose Matching Methods;
Result = Run matchTemplate(templateImage,
```

```
inputImage,method);
Smooth Result;
Find matching point in Result;
Plot Matching point position;
}
```

Глава III. Разработка приложений для распознавания образов

3.1. Используемые инструменты разработки

Android Studio. Android IDE - среда разработки под Android, основанная на Eclipse. Предоставляет интегрированные инструменты для разработки, сборки и отладки мобильных приложений. В данном курсе Android IDE выбрана в качестве основной среды разработки. Возможности этой среды более подробно рассмотрены в первой лабораторной работе. Также там даны рекомендации по установке и настройке среды, созданию и запуску первого приложения как на эмуляторе, так и на реальном устройстве.

Android Studio - среда разработки под Android, основанная на IntelliJ IDEA. Подобно Android IDE, она предоставляет интегрированные инструменты для разработки и отладки. Дополнительно ко всем возможностям, ожидаемым от IntelliJ, в Android Studio реализованы:

- поддержка сборки приложения, основанной на Gradle;
- специфичный для Android рефакторинг и быстрое исправление дефектов;
- lint инструменты для поиска проблем с производительностью, с юзабилити, с совместимостью версий и других;
- возможности ProGuard (утилита для сокращения, оптимизации и обфускации кода) и подписи приложений;
- основанные на шаблонах мастера для создания общих Android конструкций и компонентов;
- WYSIWYG редактор, работающий на многих размерах экранов и разрешений, окно предварительного просмотра, показывающее

запущенное приложение сразу на нескольких устройствах и в реальном времени;

- встроенная поддержка облачной платформы Google.

Установка Android Studio. Первое, что понадобиться нам для установки Android Stuido — это скачать его дистрибутив с официального сайта https://developer.android.com/studio.

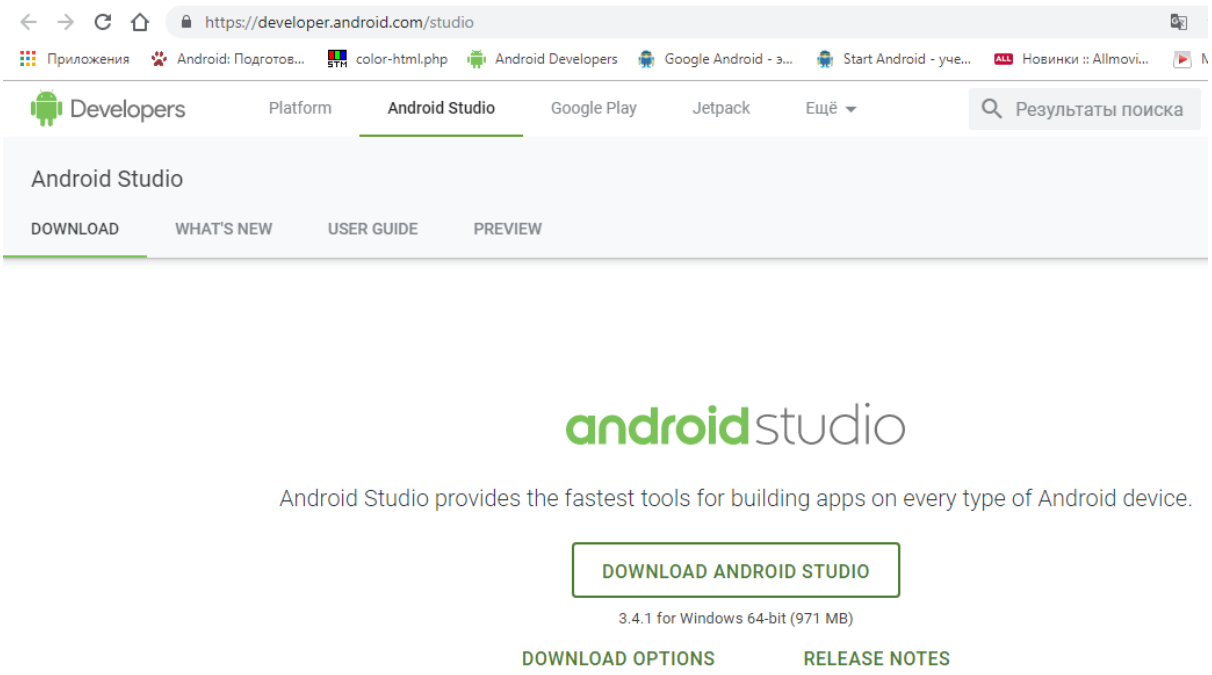


Рисунок 3.1. Скачивание дистрибутива Android Studio

Далее мы должны согласиться с публичной офертой лицензионного соглашение для Android Software Development Kit

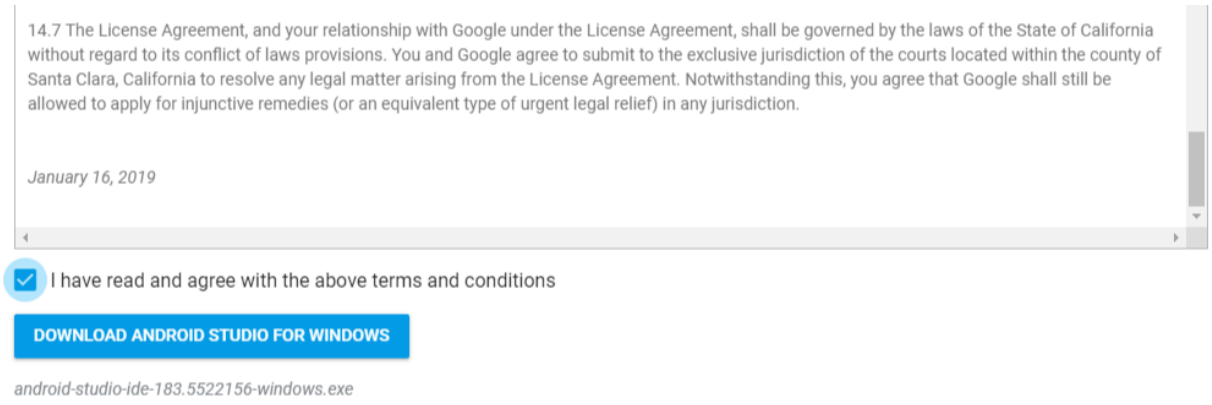


Рисунок 3.2. Соглашение о скачивании программы

Полученный результат после скачивания программы с официального сайта.

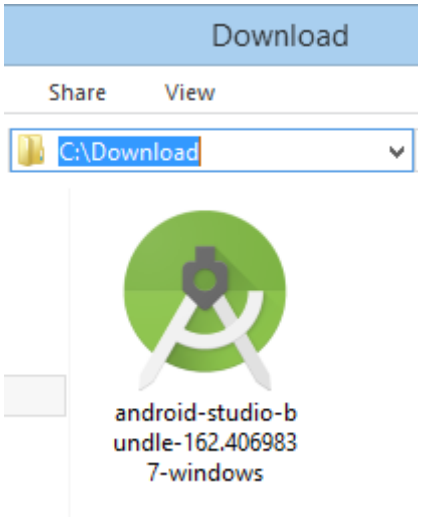


Рисунок 3.3. Скаченный дистрибутив Android Studio

После того, как мы скачали дистрибутив программы для дальнейшей ее работы на понадобиться Java Development Kit. Java Development Kit (сокращенно *JDK*) — бесплатно распространяемый компанией *Oracle Corporation* комплект разработчика приложений на языке *Java*, включающий в себя компилятор *Java*, стандартные библиотеки классов *Java*, примеры, документацию, различные утилиты и исполнительную систему *Java*. В состав *JDK* не входит интегрированная среда разработки на *Java*, поэтому разработчик, использующий только *JDK*, вынужден использовать внешний текстовый редактор и компилировать свои программы, используя утилиты командной строки. Java Development Kit я скачал с его официального сайта https://www.oracle.com/technetwork/java/javase/downloads/index.html.

Рисунок 3.4. Java Development Kit

Теперь у меня собраны все компоненты для установки среды Android Studio. Я в первую очередь установлю компонент Java Development Kit, после начну установку среды Android Studio.

Рисунок 3.5. Начало установки Android Studio

Выбираем все опции (options).

The Android SDK (комплект средств разработки) это комплект инструментов использующихся для разработки приложений Android. Android SDK включает:

Отладчик (Debugger)

Эмулятор (Emulator)

Связанные документы для Android API.

Пример кода.

Руководство для Android.

Android Virtual Device (AVD) это устройство конфигурации запускающий эмулятор Android (Android emulator). Оно работает с эмулятором чтобы предоставить определенную виртуальную среду, чтобы установить и запустить приложение Android.

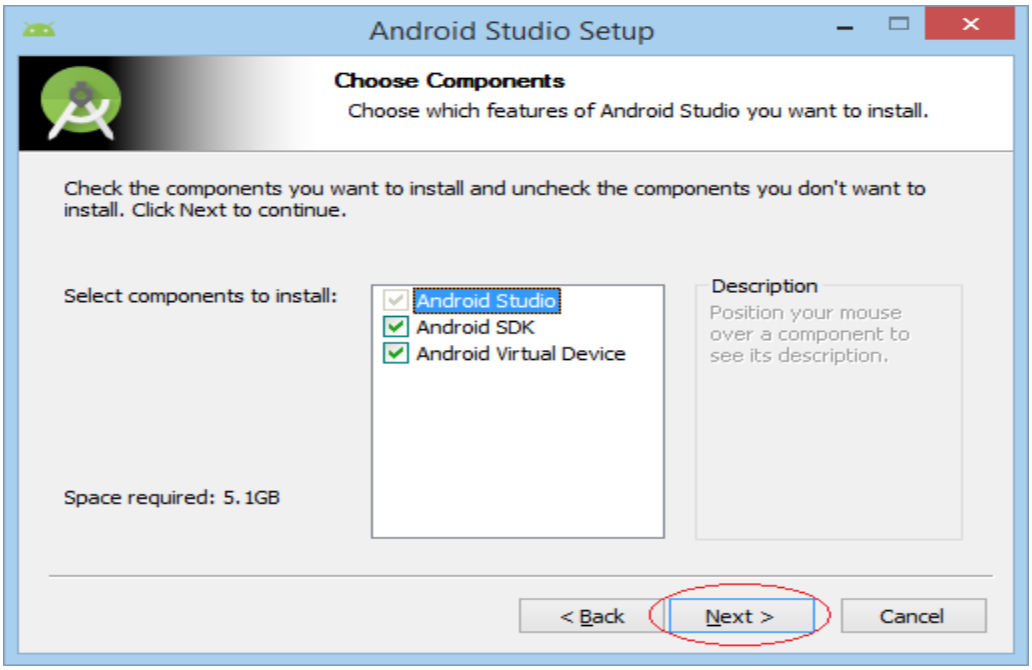


Рисунок 3.6. Выборка компонентов

В этом окне я выбираю все компоненты для установки.

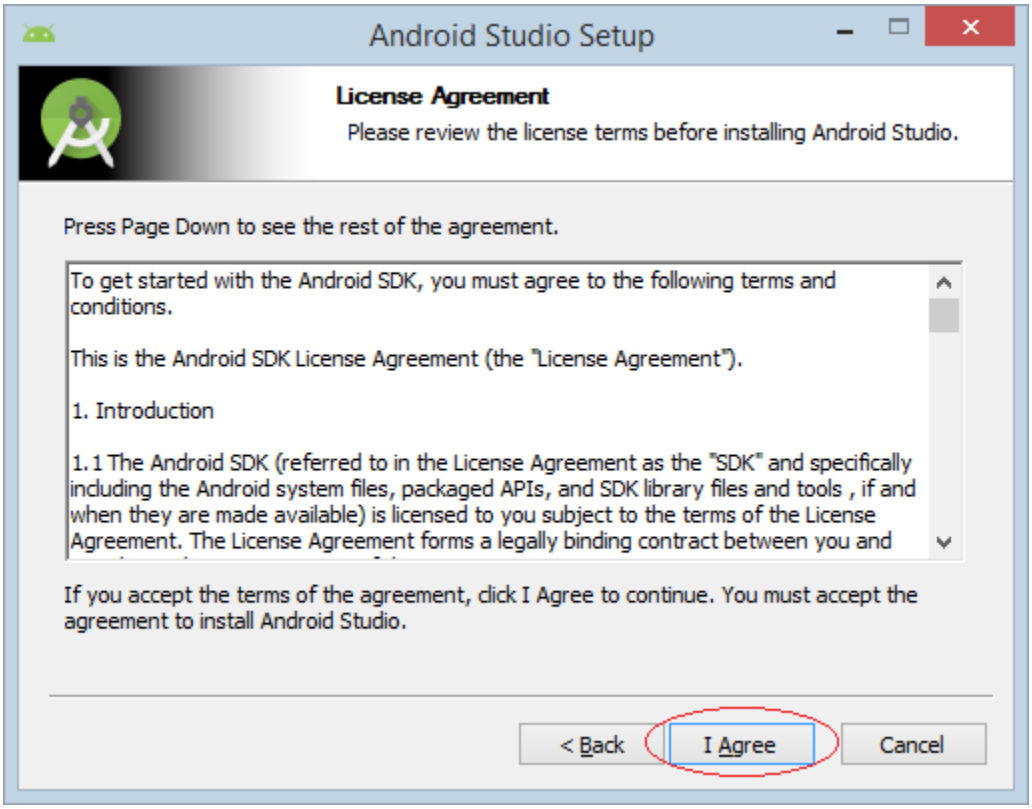


Рисунок 3.7. Соглашение

В этом открывшемся окне мы соглашаемся с условиями установки программного продукта от компании Google.

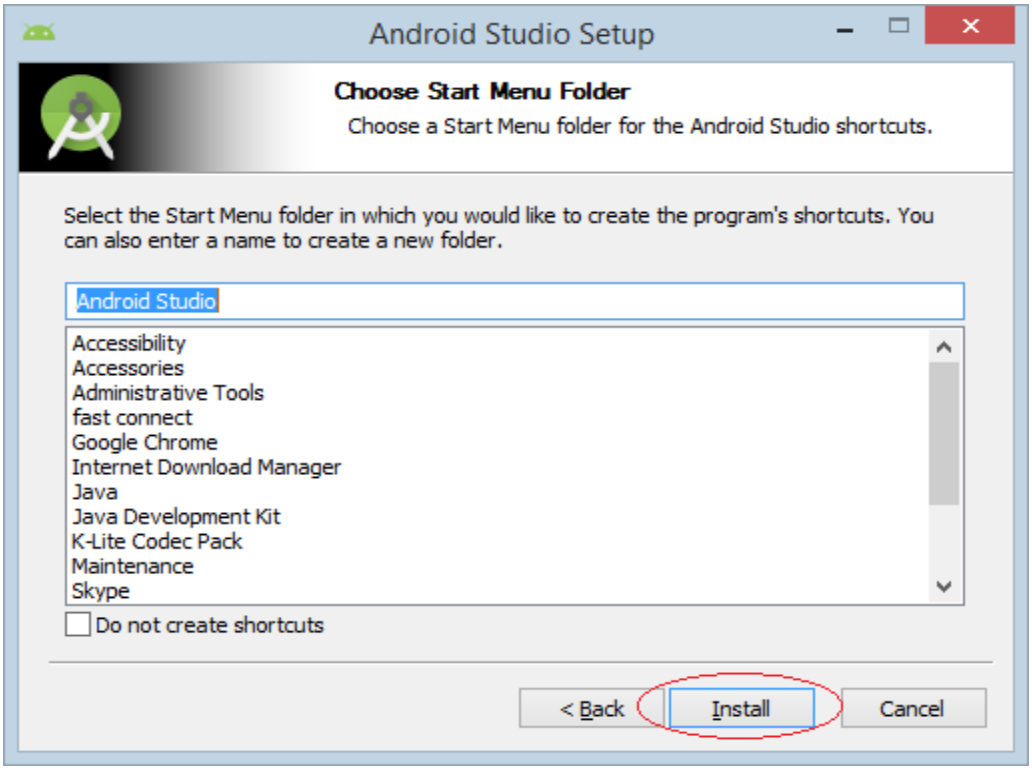


Рисунок 3.8. Выбор папки запуска

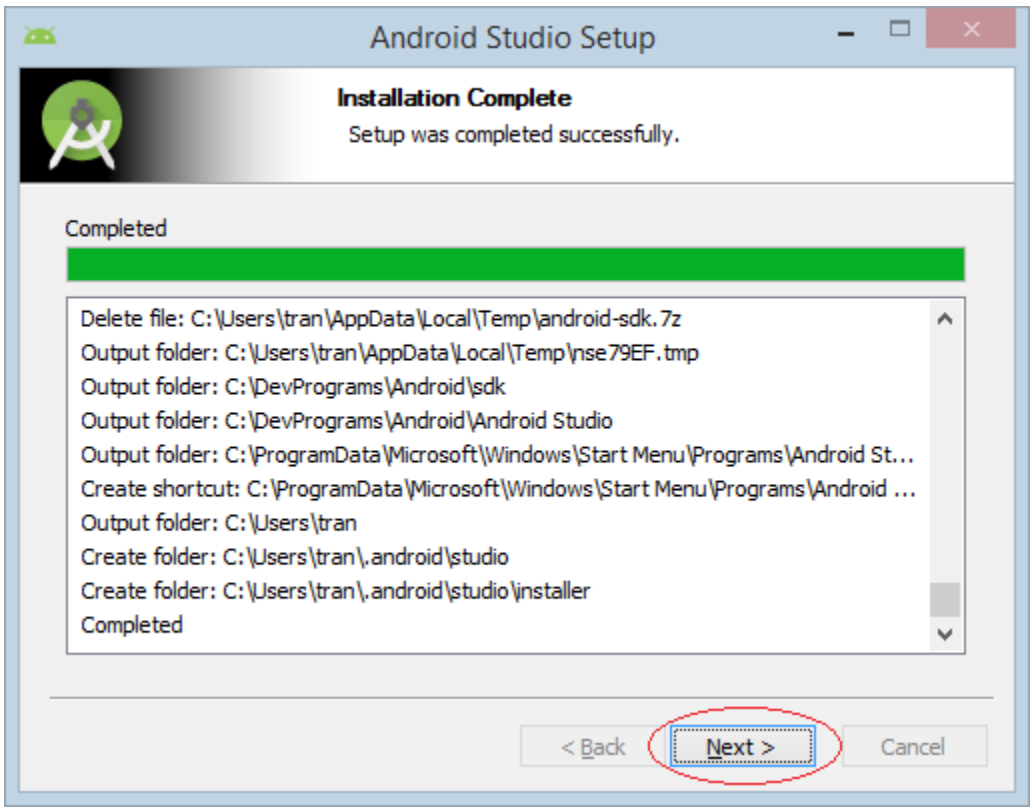


Рисунок 3.9. Завершение распаковки файлов дистрибутива

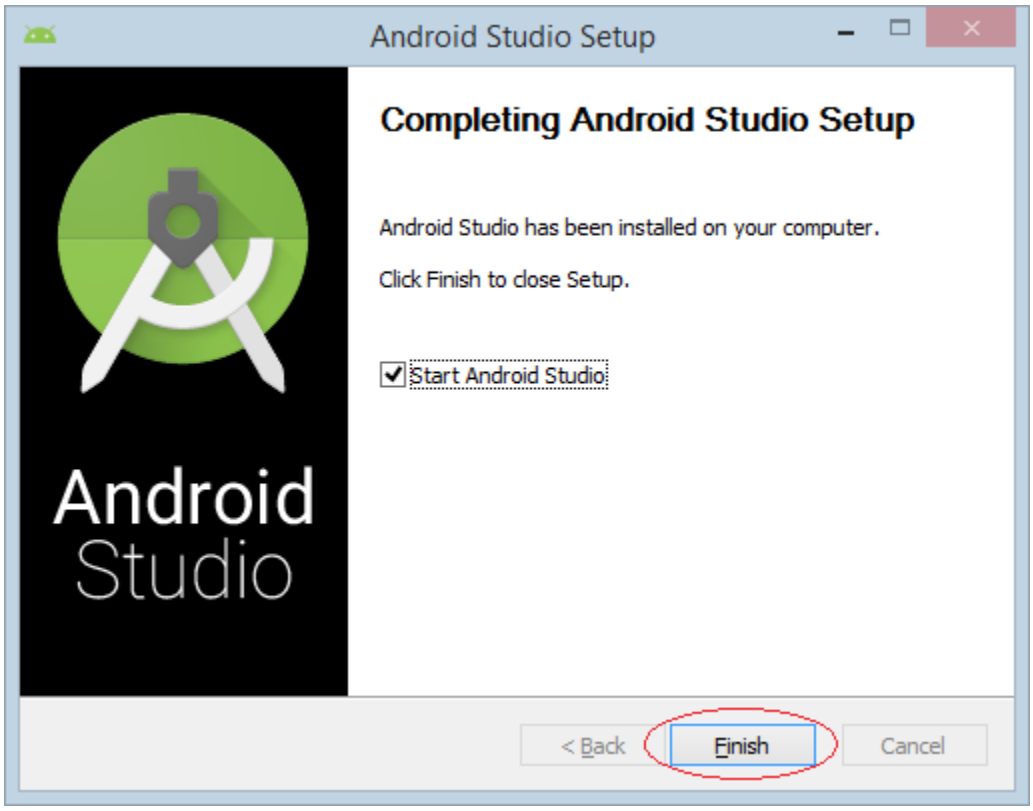


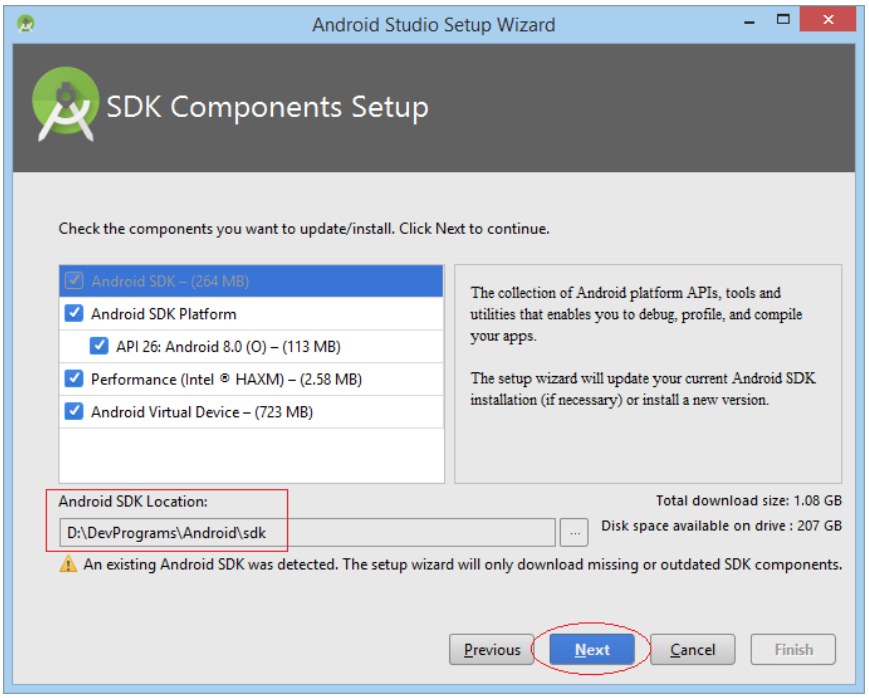
Рисунок 3.10. Завершение установки дистрибутива

Вот и последнее окно в котором завершается установка программного продукта Android Studio от компании Google.

SDK (от англ. *software development kit*). Набор средств разработки, который позволяет специалистам по программному обеспечению создавать приложения для определённого пакета программ, программного обеспечения базовых средств разработки, аппаратной платформы, компьютерной системы, игровых консолей, операционных систем и прочих платформ.

Программист, как правило, получает *SDK* непосредственно от разработчика целевой технологии или системы. Часто *SDK* распространяется через Интернет. Многие *SDK* распространяются бесплатно для того, чтобы побудить разработчиков использовать данную технологию или платформу.

Далее я показал, как установить Software development kit на уже нами установленном дистрибутиве Android Studio. Я скачал их с помощью подпрограммы SDK Manager который входит в состав Android Studio.



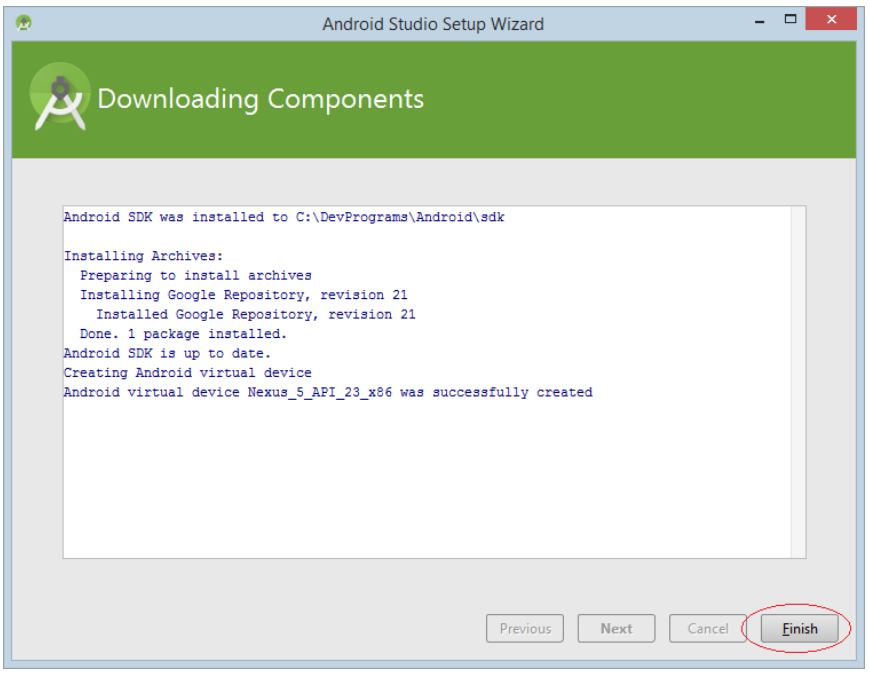


Рисунок 3.11. Установка и скачивание компонентов SDK

Библиотека OpenCV. OpenCV (англ. Open Source Computer Vision Library, библиотека компьютерного зрения с открытым исходным кодом) — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++.

OpenCV предназначена для повышения вычислительной эффективности процедур обработки видеоизображения с особым упором на применение в задачах реального времени.

OpenCV написана на C хорошо оптимизирована и может использовать преимущества многоядерных процессоров. Для более полного использования возможностей библиотеки рекомендуется установить Intel Performance Primitives (IPP). Это позволит повысить производительность процедур библиотеки.

Позволяет достаточно быстро и эффективно реализовывать сложные алгоритмы машинного зрения. Библиотека содержит более 500 функций, которые позволяют реализовывать приложения работающие во многих областях, в том числе:

- контроль качества выпускаемой продукции;

- обработке изображений в медицине;

- обеспечении безопасности;

- интерфейсе пользователя;

- робототехнике.

OpenCV содержит библиотеку общих функций искусственного интеллекта «Machine Learning Library» (MLL). Она служит, в основном, для распознавания фрагментов изображения и кластеризации.

Данную библиотеку применяют:

- для утверждения общего стандартного интерфейса компьютерного зрения для приложений в этой области;

- для способствования росту числа таких приложений и создания новых моделей использования PC;

- сделать платформы Intel привлекательными для разработчиков таких приложений за счёт дополнительного ускорения OpenCV с помощью Intel® Performance Libraries (Сейчас включают IPP (низко-уровневые библиотеки для обработки сигналов, изображений, а также медиа-кодеки) и MKL (специальная версия LAPACK и FFTPack));

- OpenCV способна автоматически обнаруживать присутствие IPP и MKL и использовать их для ускорения обработки.

Основными модулями являются следующие модули.

Ядро *cxcore* производит:

- базовые операции над многомерными числовыми массивами;

- матричная алгебра, математические функции, генераторы случайных чисел DFT, DCT;

- запись/восстановление структур данных в/из XML/YAML;

- базовые функции 2D графики;

- поддержка более сложных структур данных: разреженные массивы, динамически растущие последовательности, графы;

CV — является модулем обработки изображений и компьютерного зрения. Его функциями являются:

- базовые операции над изображениями (фильтрация, геометрические преобразования, преобразование цветовых пространств и т. д.)
- анализ изображений (выбор отличительных признаков, морфология, поиск контуров, гистограммы);
- структурный анализ (описание форм, плоские разбиения);
- анализ движения, слежение за объектами;
- обнаружение объектов, в частности лиц;
- калибровка камер, элементы восстановления пространственной структуры;

Highgui — Модуль для ввода/вывода изображений и видео, создания пользовательского интерфейса

Необходим для выполнения следующих операций:

- захват видео с камер и из видео файлов, чтение/запись статических изображений;
- функции для организации простого UI (сейчас все демо приложения используют HighGUI).

Cvaux — Экспериментальные и устаревшие функции

Данный модуль выполняет:

- пространственное зрение: стерео калибрация, само калибрация;
- поиск стерео-соответствия, клики в графах;
- нахождение и описание черт лица;
- сравнение форм, построение скелетонов;
- скрытые Марковские цепи;
- описание текстур.

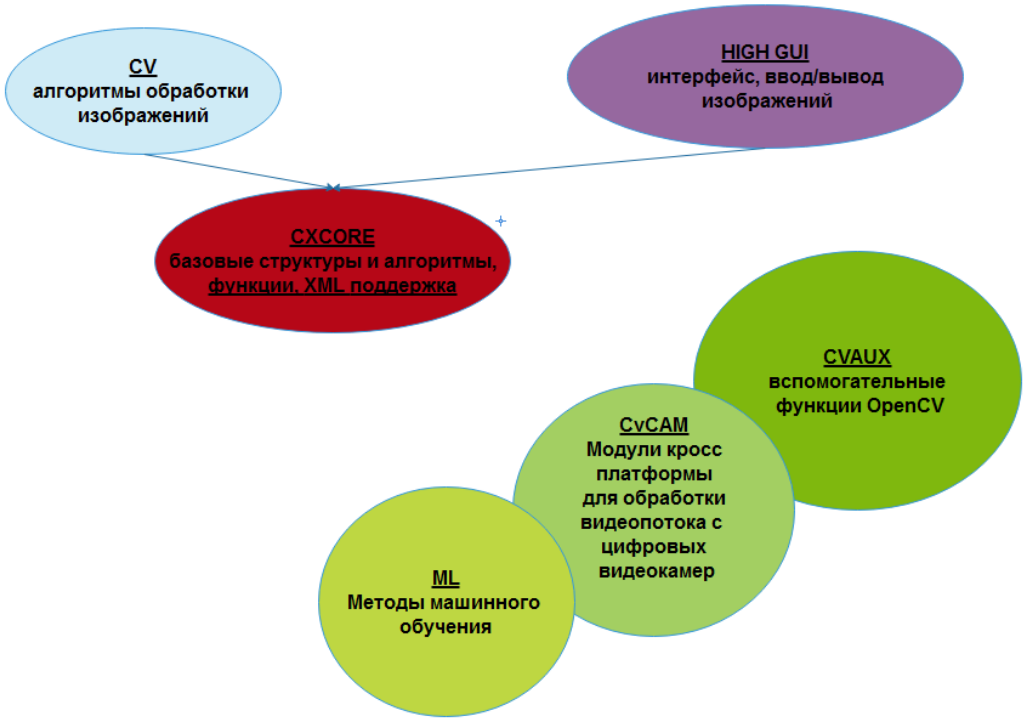


Рисунок 3.12. Описание модулей

Библиотека Android Vision. *Представьте, вам надо оцифровать журнальную статью или распечатанный договор. Конечно, вы можете провести несколько часов, перепечатывая документ и исправляя опечатки. Либо вы можете перевести все требуемые материалы в редактируемый формат за несколько минут, используя сканер (или цифровую камеру) и программу для оптического распознавания символов.*

Оптическое распознавание символов (англ. Optical Character Recognition – OCR) – это технология, которая позволяет преобразовывать различные типы документов, такие как отсканированные документы, PDF-файлы или фото с цифровой камеры, в редактируемые форматы с возможностью поиска.

Изображение, согласно принципу целостности, будет интерпретировано как некий объект, только если на нем присутствуют все структурные части этого объекта и эти части находятся в соответствующих отношениях. Иначе говоря, приложение не пытается принимать решение, перебирая тысячи эталонов в поисках наиболее подходящего. Вместо этого выдвигается ряд

гипотез относительно того, на что похоже обнаруженное изображение. Затем каждая гипотеза целенаправленно проверяется. И, допуская, что найденный объект может быть буквой А, приложение с помощью библиотеки будет искать именно те особенности, которые должны быть у изображения этой буквы. Как и следует поступать, исходя из принципа целенаправленности. Принцип адаптивности означает, что приложение должна быть способна к самообучению, поэтому проверять, верна ли выдвинутая гипотеза, система будет, опираясь на накопленные ранее сведения о возможных начертаниях символа в данном конкретном документе.

Распознавание структуры текста. Распознаватель текста разделяет текст на блоки, строки и слова. Грубо говоря:

- Блок - это непрерывный набор текстовых строк, таких как абзац или столбец;
- Линия - это непрерывный набор слов на одной вертикальной оси;
- Слово - это непрерывный набор буквенно-цифровых символов на одной вертикальной оси.

На рисунке 3.13. ниже показаны примеры каждого из них в порядке убывания. Первый выделенный блок на голубом - это блок текста. Второй набор выделенных блоков синим цветом - это строки текста. Наконец, третий набор выделенных блоков, темно-синего цвета, - это слова.

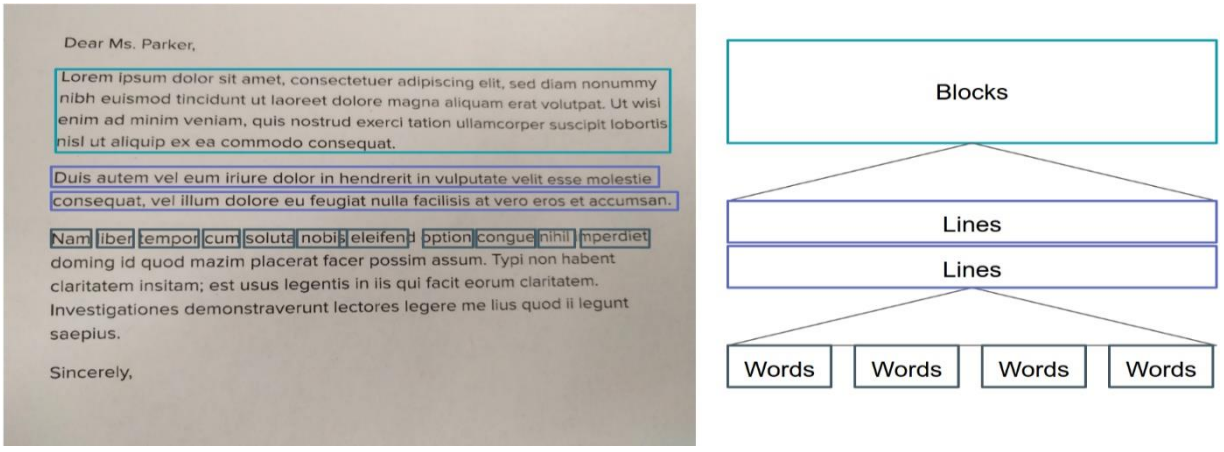


Рисунок 3.13. Работа библиотеки OCR

3.2. Этапы разработки приложений

На основе выбранной мною темы, я попытаюсь показать вам проделанною мною работу о разработке двух приложений на платформе Android, которые непосредственно связаны с библиотеками компьютерного зрения, таких как OpenCV и Optical Character Recognition.

Так как библиотека OCR уже интегрирована в Android Studio, нам не понадобиться ее скачивать, но мы должны первоочередно скачать и интегрировать библиотеку OpenCV. Библиотеку OpenCV я скачал с официального сайта https://opencv.org/releases/. Я в своих приложениях использовал версию библиотеки «*OpenCV 3.4.3 – android sdk*».

Интеграция библиотеки OpenCV в Android Studio. Первым делом я распаковал архив с библиотекой, как показано на рисунке 3.14.

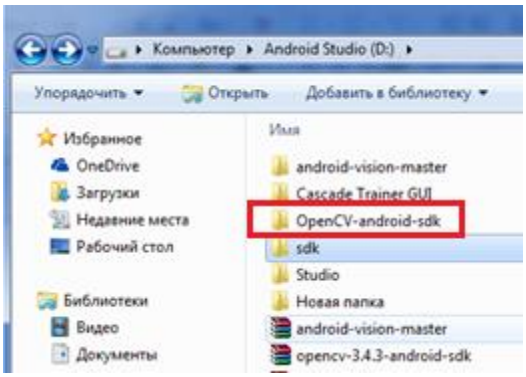


Рисунок 3.14. Распаковка библиотеки OpenCV

Создаем проект в Android Studio под именем «Diploma Kazakbaev S.», устанавливая минимальную версии Android устройства «Lollilop 5.1».

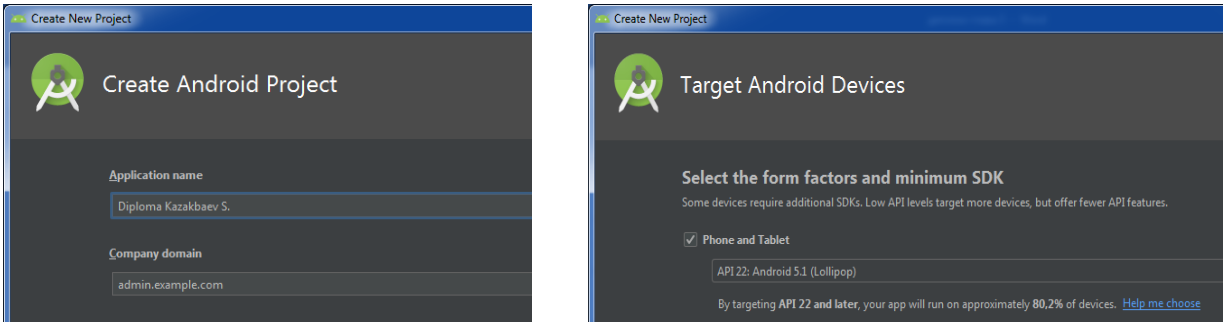


Рисунок 3.15. Создание проекта

После успешного создания проекта Android пришло время импортировать модуль OpenCV проект Android. Нажатием File -> New -> Import Module, как показано на рисунке 3.16.

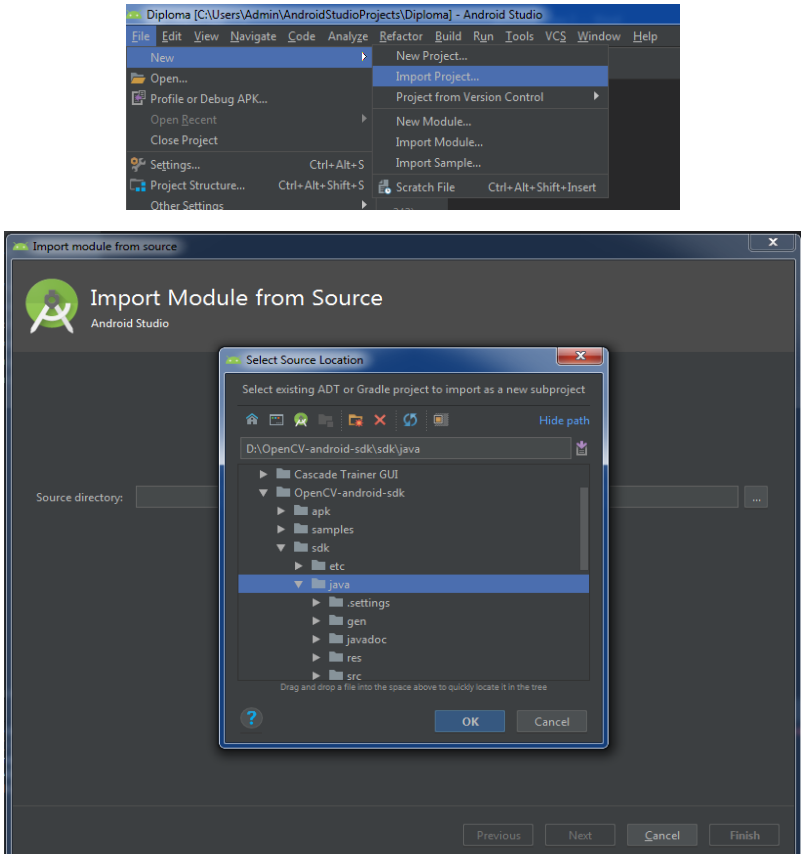


Рисунок 3.16. Импорт модуля OpenCV

Переходим в папку, в которую я ранее распаковал содержимое zip-файла библиотеки OpenCV Android. Выберя папку Java внутри папки SDK. После выбора правильного пути и нажатия на кнопу OK, я получаю успешную интеграцию модуля, как выражено на рисунке 3.17.

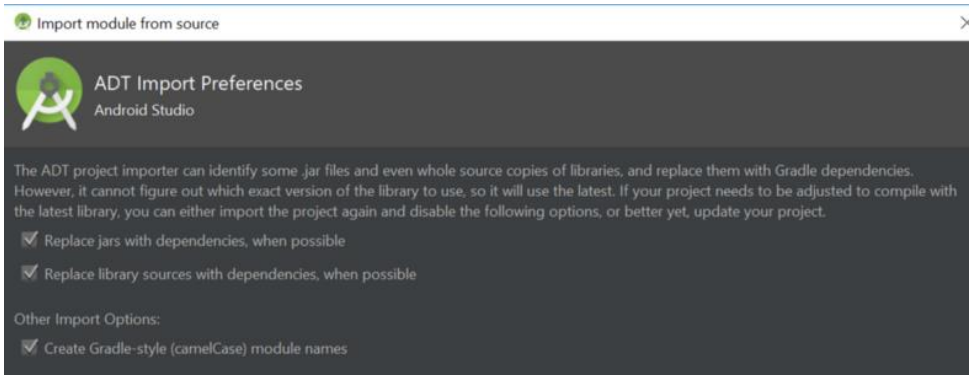


Рисунок 3.17. Результат импорта модуля OpenCV

После завершения импорта библиотеки OpenCV я получил ошибку сборки Gradle. Это происходит потому, что библиотека использует старый Android SDK, который я, вероятно, еще не установили. Решением этой проблемы будет изменение настроек Gradle. Заменяем compileSdkVersion и targetSdkVersion на последнюю версию Android SDK, которая установлена на моем ПК. После изменения версии запускаем синхронизацию проекта, чтобы Gradle смог синхронизироваться с модулем, который описывается на рисунке 3.18.

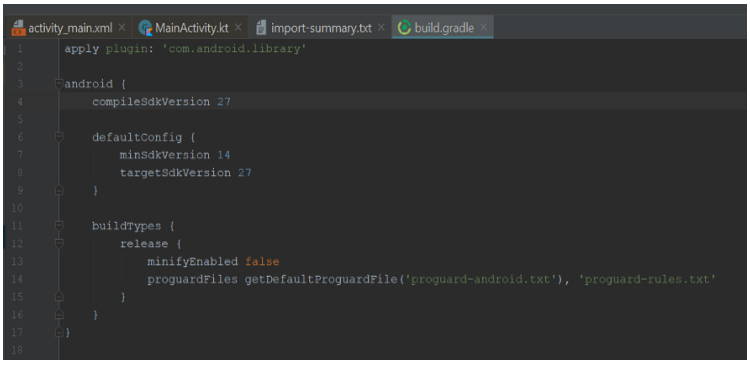


Рисунок 3.17. Настройка Gradle

Следующим шагом будет являться добавление зависимости проекта от библиотеки OpenCV — File -> Project Structure. Когда откроется диалоговое окно структуры проекта, нажимаем на кнопку модуль приложения или любой другой модуль, в котором вы хотите использовать библиотеку OpenCV. После перехода к модулю, нажмите на вкладку Зависимости. Нажимая на зеленую кнопку «плюс» в дальнем правом углу диалогового окна, и выбраю «Зависимость модуля».

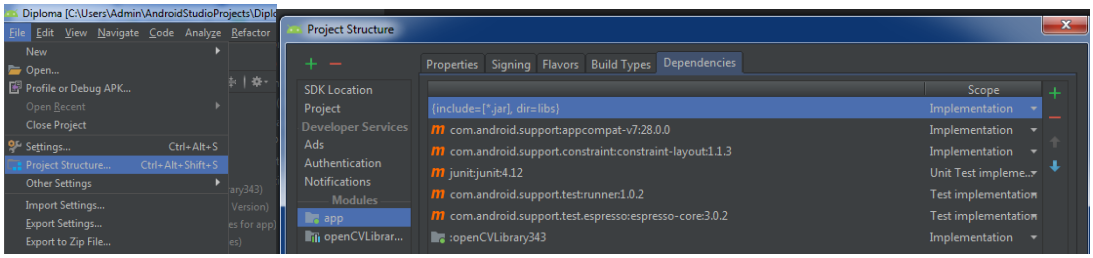


Рисунок 3.18. Добавление зависимости от OpenCV

Предпоследним шагом для работы модуля OpenCV будет являться копирование библиотек из распакованного каталога в каталог проекта. Копируем каталог OpenCV-android-sdk/sdk/native/libs в каталог проекта Diploma Kazakbaev S./app/src/main и переименовываем каталог libs в jniLibs.

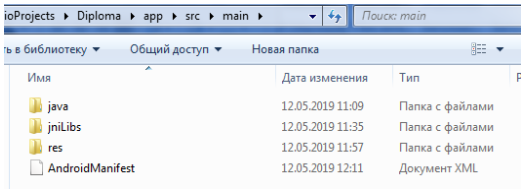


Рисунок 3.19. Копирование библиотек

Заключающим шагом исполнителем является изменение доступа камеры приложением в файле AndroidManifest.xml, как показано в таблице 3.1.

Таблица 3.1. Изменение параметров камеры

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.admin.diploma">
   <uses-permission android:name="android.permission.CAMERA"/>
   <uses-permission android:name="android.permission.CAMERA"/>
   <uses-feature android:name="android.hardware.camera"
android:required="false"/>
   <uses-feature android:name="android.hardware.camera.autofocus"
android:required="false"/>
   <uses-feature android:name="android.hardware.camera.front"
android:required="false"/>
   <uses-feature android:name="android.hardware.camera.front.autofocus"
android:required="false"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
```

```
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">          <intent-filter>
              <action android:name="android.intent.action.MAIN" />
              <category android:name="android.intent.category.LAUNCHER" />
           </intent-filter>
        </activity>
    </application>
</manifest>
```

Ну что же, завершив все установки дополнительных компонентов, которое требует на стабильной работы нашего приложения, напишем программный код, который будет детектировать лица, и производить поиск текста. Так как мы будем работать с камерой в реальном времени, нам понадобиться весь экран мобильного устройства. Эти параметры мы изменим в *activity_main.xml,* как показано в таблице 3.2.

Таблица 3.2. Настройка параметром экрана

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:opencv="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <org.opencv.android.JavaCameraView
        android:id="@+id/view"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:visibility="gone"
```

```
    opencv:camera_id="any" />
</FrameLayout>
```

Для детектирования лиц был использован готовый каскад *lbpcascade_frontalface.xml* из библиотеки OpenCV, который я расположил в директории *raw* Android проекта, его код будет показан в приложение 1. Далее следовала дать, такие параметры приложению, чтобы оно всегда проверяла активность библиотеки OpenCV. Код был написан в файле *MainActivity.java* в директории *Java*.Код приведен ниже в таблице 3.3.

Таблица 3.3. Проверка подключения OpenCV в приложении

```
private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this)
{
    @Override
    public void onManagerConnected(int status) {
        switch (status) {
            case LoaderCallbackInterface.SUCCESS:
                initializeOpenCVDependencies();
                break;
            default:
                super.onManagerConnected(status);
                break;
        }
    }
};

private void initializeOpenCVDependencies() {

    try {
```

```
        InputStream is = getResources().openRawResource(R.raw.lbpcascade_frontalface);
        File cascadeDir = getDir("cascade", Context.MODE_PRIVATE);
        File mCascadeFile = new File(cascadeDir, "lbpcascade_frontalface.xml");
        FileOutputStream os = new FileOutputStream(mCascadeFile);
        byte[] buffer = new byte[4096];
        int bytesRead;
        while ((bytesRead = is.read(buffer)) != -1) {
           os.write(buffer, 0, bytesRead);
        }
        is.close();
        os.close();
        cascadeClassifier = new CascadeClassifier(mCascadeFile.getAbsolutePath());
      } catch (Exception e) {
        Log.e("OpenCVActivity", "Error loading cascade", e);
      }
      openCvCameraView.enableView();    }
```

Далее стоит отметить что, нам нужно выделить только часть тело человека, то есть определить его лицо в режиме реального времени. Для этого создаем класс *onCreate*, котором мы берём параметры экрана мобильного устройства. Также создаем класс *onCameraViewStarted*, передавая ей параметры экрана мобильного устройства, объявляем в ней переменную Mat, которая используя алгоритм из библиотеки OpenCV, будет находить лица с помощью каскада *lbpcascade_frontalface.xml*. Каскад будет определять есть ли определенные лица в установленном положении камеры, и будет заполнять оперативную память устройства, в противном случая нет. Если каскад определит, что на поле досягаемости камеры мобильного устройства объявилось лицо, то массив *Rect[]* используя функцию Imgproc.rectangle()

отрисует прямоугольник вокруг лица субъекта или же субъектов красным цветом. Код размещен ниже:

Таблица 3.4. Детектирование с помощью каскада и отрисовка прямоугольник вокруг лиц

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREE
N_ON);
        openCvCameraView = new JavaCameraView(this, -1);
        setContentView(openCvCameraView);
        openCvCameraView.setCvCameraViewListener(this);
    }
    @Override
    public void onCameraViewStarted(int width, int height) {
        grayscaleImage = new Mat(height, width, CvType.CV_8UC4);
        absoluteFaceSize = (int) (height * 0.2);
    }
    @Override
    public void onCameraViewStopped() {
    }
    @Override
    public Mat onCameraFrame(Mat aInputFrame) {
        Imgproc.cvtColor(aInputFrame, grayscaleImage,
Imgproc.COLOR_RGBA2RGB);
        MatOfRect faces = new MatOfRect();
        if (cascadeClassifier != null) {
            cascadeClassifier.detectMultiScale(grayscaleImage, faces, 1.1, 2, 2,
```

```
                new Size(absoluteFaceSize, absoluteFaceSize), new Size());
        }
        Rect[] facesArray = faces.toArray();
        for (int i = 0; i <facesArray.length; i++)
            Imgproc.rectangle(aInputFrame, facesArray[i].tl(), facesArray[i].br(),
 new Scalar(0, 255, 0, 0), 3);
        return aInputFrame;
    }
    @Override
    public void onResume() {
        super.onResume();
        OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_4_0,
 this, mLoaderCallback);
    }
```

Вот так непосильным трудом и старанием можно воссоздать приложение для детектирование лиц на платформе Android. Ну что же, давайте не будет останавливаться на этом и рассмотри еще одно приложение, которое мы попробуем создать с библиотеки OCR — для распознавания латинских букв и цифр в режиме реального времени с помощью обычной камеры мобильного устройства.

Сохраняем параметры:

- *Gradle* как показано на рисунке 3.17
- *AndroidManifest.xml* и *activity_main.xml* как было показано выше в таблицах 3.1 и 3.2.

Используя готовый код оптического распознавания текста от Google Vision, я создам три Java файла под названиями:

- OcrCaptureActivity;
- OcrDetectorProcessor;

- OcrGraphic;

которые в дальнейшем будут использовать для распознавания текста. Я не стал изменять код для увеличения зума, расшумовки и стабилизации камеры, так не нужно создавать велосипед дважды, эти исходники были созданы основателями кодов камеры, которые используются во всех приложениях на Android.

Дабы приложение заработало я воссоздал переменную TextRecognizer из OCR. Это такой объект-детектор обрабатывающий изображение и определяющее, какой текст появляется внутри них. Для этого нам понадобится перейти в класс OcrCaptureActivity и методе createCameraSource добавить переменную. После я добавил условие проверки, работает ли TextRecognizer в методе createCameraSource. Далее в этом же методе я добавлю автофокусировку камеры, чтобы можно было распознавать текст в режиме реального времени. Код приведен ниже в таблице 3.5.

Таблица 3.5. Создание объекта-детектора

```
private void createCameraSource(boolean autoFocus, boolean useFlash) {
    Context context = getApplicationContext();
    TextRecognizer textRecognizer = new
TextRecognizer.Builder(context).build();
  TextRecognizer.setProcessor(new OcrDetectorProcessor(graphicOverlay));
    if (!textRecognizer.isOperational()) {
        Log.w(TAG, "Detector dependencies are not yet available.");
        IntentFilter lowstorageFilter = new
IntentFilter(Intent.ACTION_DEVICE_STORAGE_LOW);
        boolean hasLowStorage = registerReceiver(null, lowstorageFilter) != null;
        if (hasLowStorage) {
            Toast.makeText(this, R.string.low_storage_error,
Toast.LENGTH_LONG).show();
```

```
            Log.w(TAG, getString(R.string.low_storage_error));
        }
    }
    cameraSource =
            new CameraSource.Builder(getApplicationContext(), textRecognizer)
            .setFacing(CameraSource.CAMERA_FACING_BACK)
            .setRequestedPreviewSize(1280, 1024)
            .setRequestedFps(15.0f)
            .setFlashMode(useFlash ?
 Camera.Parameters.FLASH_MODE_TORCH : null)
            .setFocusMode(autoFocus ?
 Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO : null)
            .build();
 }
```

Теперь если приложение запустить, то оно заработает, но детектирование текста не будет работать, так как еще нужно добивать процессор детектирования. Для этого переходим в OcrDetectorProcessor и реализуем интерфейс Detector.Processor. Для реализации этого интерфейса требуется переопределить два метода. Первый, receiveDetections, который получает на вход TextBlocks из TextRecognizer по мере их обнаружения. Второй, release, используется для освобождения от ресурсов при уничтожении TextRecognizer. В этом случае нам нужно просто очистить графическое полотно, что приведёт к удалению всех объектов OcrGraphic. Мы получим TextBlocks и создадим объекты OcrGraphic для каждого текстового блока, обнаруженного процессором. Логику их рисования предопределим на следующем шаге.

Таблица 3.6. Реализация интерфейса Detector.Processor

```
public class OcrDetectorProcessor implements Detector.Processor<TextBlock> {
    private GraphicOverlay<OcrGraphic> graphicOverlay;
    OcrDetectorProcessor(GraphicOverlay<OcrGraphic> ocrGraphicOverlay) {
        graphicOverlay = ocrGraphicOverlay;
    }
    @Override
    public void receiveDetections(Detector.Detections<TextBlock> detections) {
        graphicOverlay.clear();
        SparseArray<TextBlock> items = detections.getDetectedItems();
        for (int i = 0; i < items.size(); ++i) {
            TextBlock item = items.valueAt(i);
            if (item != null && item.getValue() != null) {
                Log.d("Processor", "Text detected! " + item.getValue());
                OcrGraphic graphic = new OcrGraphic(graphicOverlay, item);
                graphicOverlay.add(graphic);
            }
        }
    }
    @Override
    public void release() {
        graphicOverlay.clear();
    }
}
```

Теперь это приложение обнаруживает текст, но я хочу пойти дальше и отобразить этот текст на экране. Для этого приложению нужно понять, есть ли текст на изображении, и для этого я добавляю метод draw в OcrGraphic. Далее при обнаружении текста, приложение я доработал таким образом чтобы оно

преобразовала координаты границ текста в рамки функции *canvas* и отрисовала текст внутри этой рамки. Код приведен ниже в таблице 3.7.

Таблица 3.7. Отображение рамки и текста на экране мобильного устройства

```
@Override
public void draw(Canvas canvas) {
    if (text == null) {
        return;
    }
    RectF rect = new RectF(text.getBoundingBox());
    rect = translateRect(rect);
    canvas.drawRect(rect, rectPaint);
    List<? extends Text> textComponents = text.getComponents();
    for(Text currentText : textComponents) {
        float left = translateX(currentText.getBoundingBox().left);
        float bottom = translateY(currentText.getBoundingBox().bottom);
        canvas.drawText(currentText.getValue(), left, bottom, textPaint);    }
}
```

На этом все, приложение реализовано, но почему меня это не удовлетворило, а давайте заставим приложение произносить распознанный текст. Для этого я воспользуемся библиотекой *TexttoSpeech API* встроенный в Android, и добавлю метод *contains* в *OcrGraphic*.

Таблица 3.8. Преобразование *contains*

```
public boolean contains(float x, float y) {
    if (text == null) {
        return false;
    }
```

```
    RectF rect = new RectF(text.getBoundingBox());
    rect = translateRect(rect);
    return rect.contans(x, y);}
```

Последними шагами осталось добавить метод *Ontap* в *OcrCaptureActivity* для обработки нажатия, дабы текст воспроизводился используя библиотеку *Texttospeech.*

Таблица 3.9. Добавление обработки нажатия *Ontap*

```
private boolean onTap(float rawX, float rawY) {
    OcrGraphic graphic = graphicOverlay.getGraphicAtLocation(rawX, rawY);
    TextBlock text = null;
    if (graphic != null) {
        text = graphic.getTextBlock();
        if (text != null && text.getValue() != null) {
            Log.d(TAG, "text data is being spoken! " + text.getValue());
            tts.speak(text.getValue(), TextToSpeech.QUEUE_ADD, null,
"DEFAULT");
        }
        else {
            Log.d(TAG, "text data is null");
        }
    }
    else {
        Log.d(TAG,"no text detected");
    }
    return text != null;}
```

Осталось последнее инициализация движка TextToSpeech для дальнейшего использования. Перейдем в метод *oncreate* в *OcrCaptureActivity* и добавим следующий фрагмент кода.

Таблица 3.10. Установка движка *TextToSpeech*

```
@Override
public void onCreate(Bundle bundle) {
   TextToSpeech.OnInitListener listener = new TextToSpeech.OnInitListener() {
         @Override
         public void onInit(final int status) {
            if (status == TextToSpeech.SUCCESS) {
               Log.d("TTS", "Text to speech engine started successfully.");
               tts.setLanguage(Locale.US);
            } else {
               Log.d("TTS", "Error starting the text to speech engine.");
            }
         }
      };
   tts = new TextToSpeech(this.getApplicationContext(), listener);}
```

3.3. Результаты работ приложений

В этой теме я хочу показать результат проделанной мною работы, который как не сказать должен показаться изящным. Давайте я продемонстрирую скриншот с рабочего стола мобильного устройства на котором уже установлены приложения. Скриншот приведен ниже на рисунке 3.20.

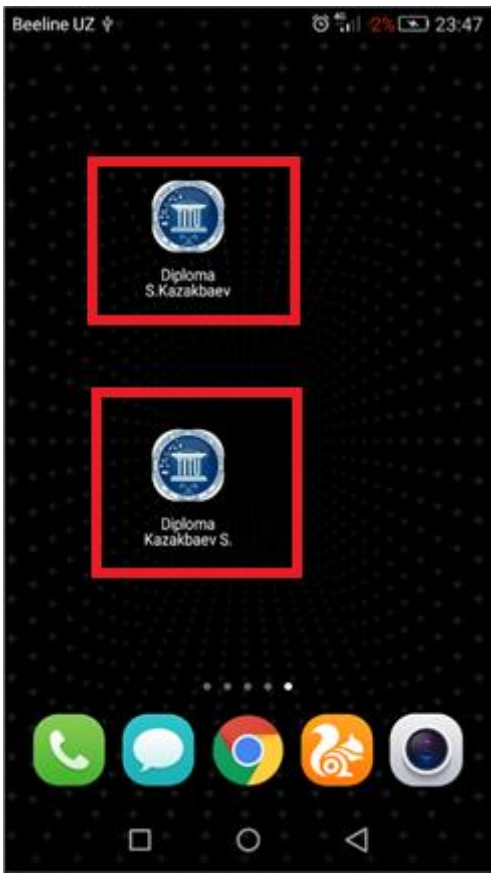


Рисунок 3.20. Скриншот икон рабочих приложений

Далее я попробую запустить приложение «*Diploma S.Kazakbaev*»— это приложение детектирования лиц разработанная на библиотеке OpenCV. Смотрим результат на рисунке 3.21.

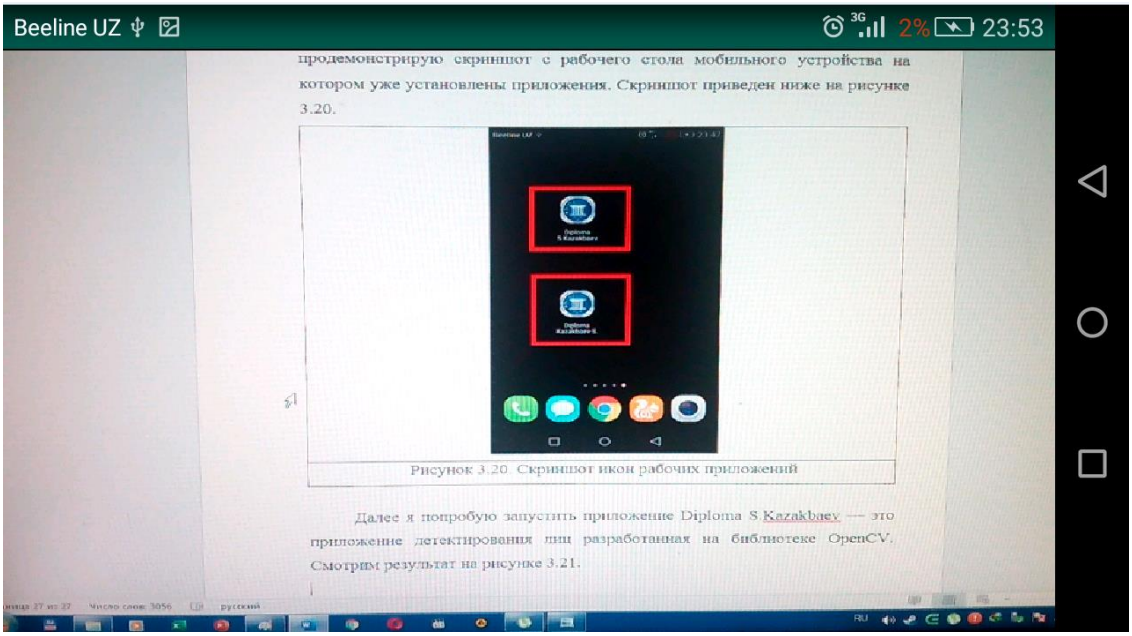


Рисунок 3.21. Запуск приложения Diploma S.Kazakbaev

Приложение запустилось, давайте теперь его протестируем, результат работы приведу ниже в рисунке 3.22.

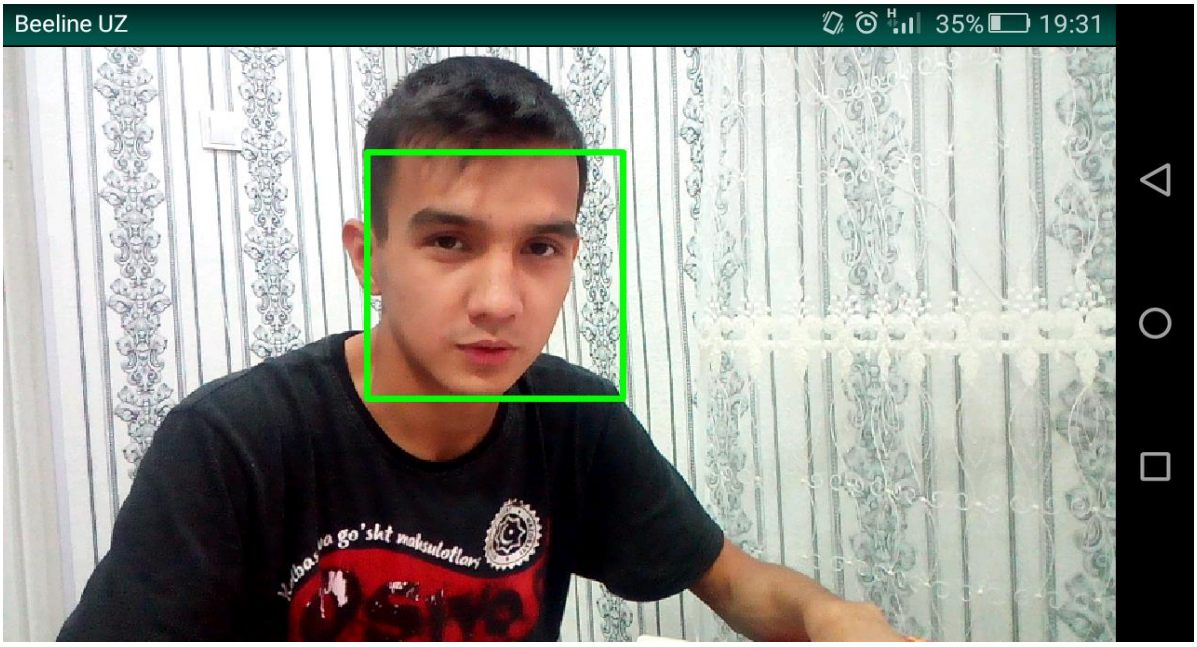


Рисунок 3.22.а. Тест приложения Diploma S.Kazakbaev

Вот и первый успешный результат приложения с добровольцем Адильбеком.

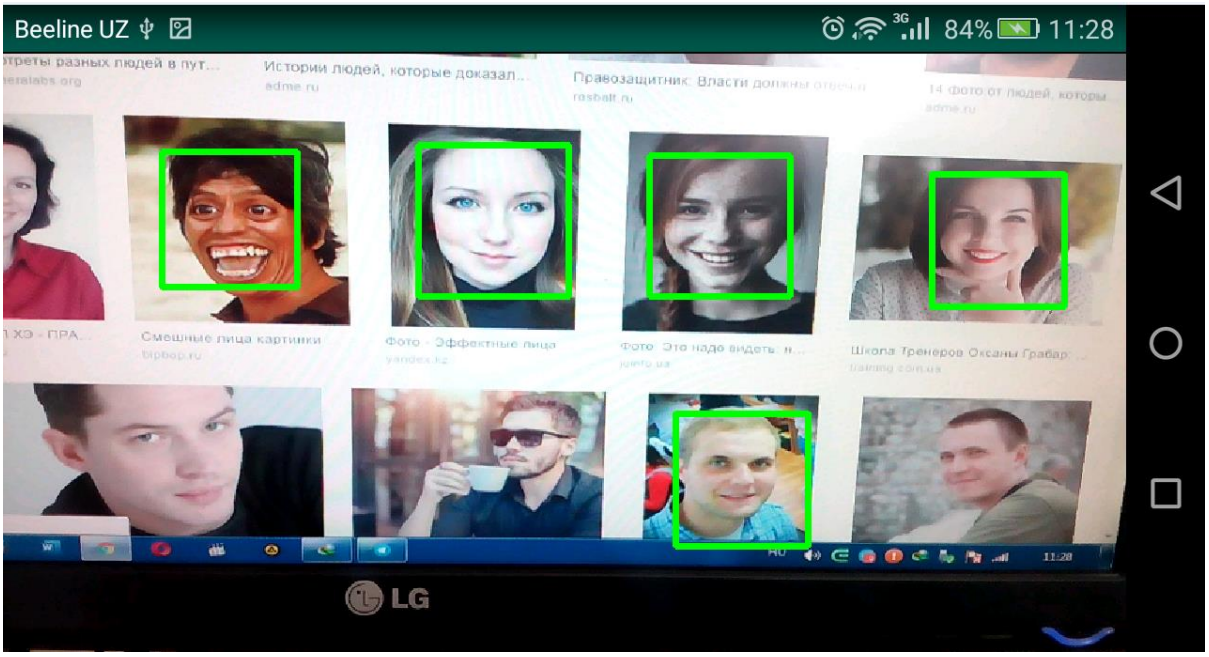


Рисунок 3.22.б. Тест приложения Diploma S.Kazakbaev

В виде объекта изображения также использовал поисковую систему Google для нахождения образом лиц людей. Как видите результат на лицо, параллельно стоящих снимков от камеры мобильного устройства.

Теперь пойдем дальше и попробуем протестировать следующее приложение «*Diploma Kazakbaev S.*», которое разработано на основе библиотеки Google Vision распознавание образов текстовой информации.

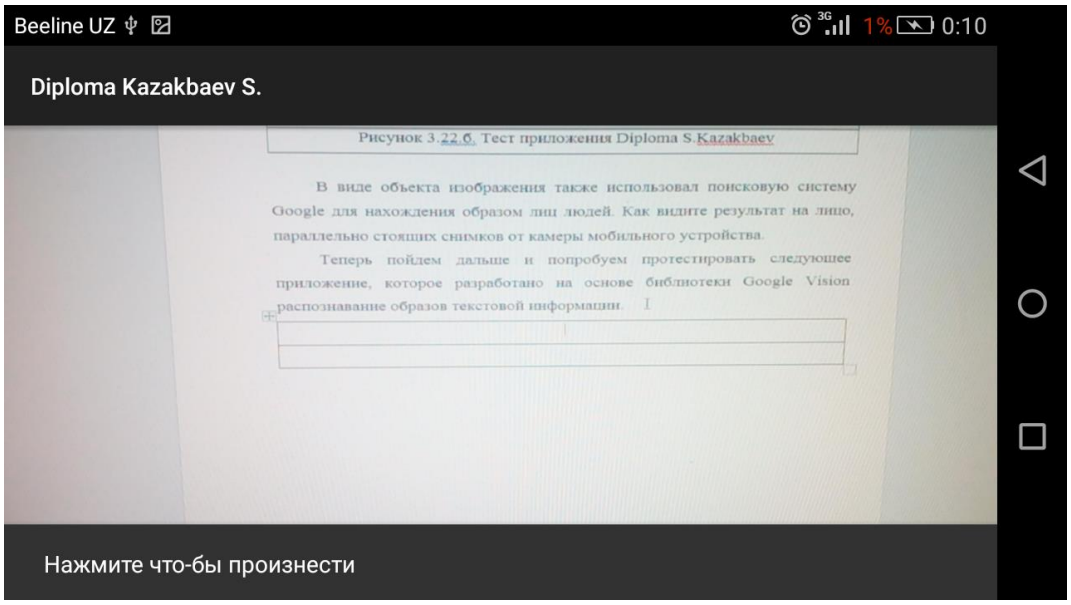
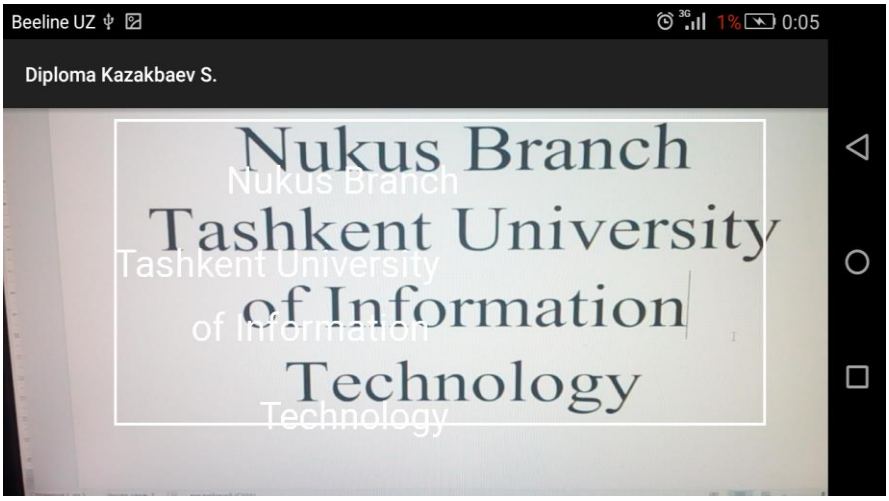


Рисунок 3.23. Запуск приложения Diploma Kazakbaev S.



(a)

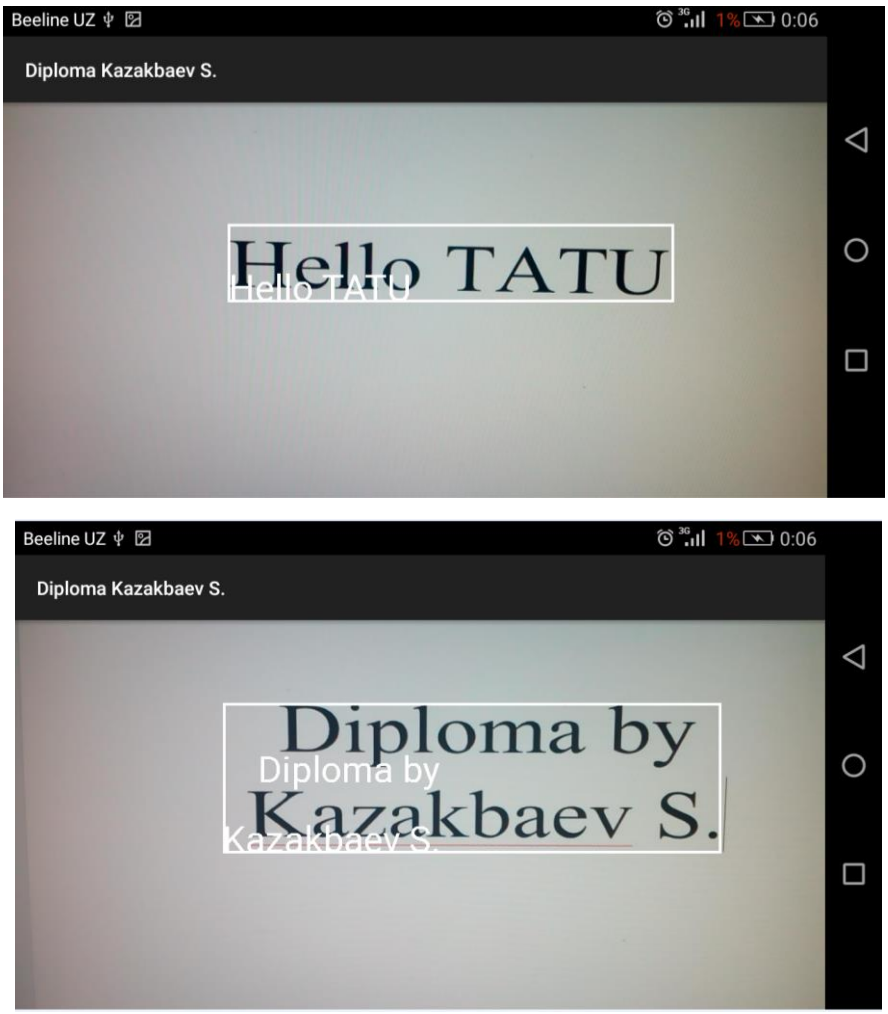


Рисунок 3.24. Тест приложения Diploma Kazakbaev S.

Как видите труды дали свои плоды, приложение работает на «хорошо», так же при нажатие на поле граница отображения мы можем услышать текст, который напечатан на изображении.

Заключение

В ходе исследования в качестве программной базы для разработки была выбрана библиотека OpenCV.

Во время проведения экспериментов было выявлено, что на точность распознавания сильно влияет качество предобработки изображения, а именно:

- выравнивание по уровню глаз;
- кадрирование;
- использование фильтров;
- выравнивание яркости и повышение контрастности.

Безусловно, существует множество направлений по развитию данных приложений. Возможно, осуществить распознавание не только цифр и других символов, но возможное осуществления программного продукта который может потягаться с AbbyyFineReader. По аналогичному алгоритму, так же существует потенциал по изменению способов распознавания символов, например, имитируя написание на бумаге или вырезания их на дереве, т.е. возможно имитировать различные текстуры и написание символов на данных текстурах различными предметами. Следовательно, программа станет ещё более привлекательной для пользователя и конкурентоспособной на современном рынке. А также приложение может быть полезна и при обучении студентов, и при разработке различного программного обеспечения для персональных компьютеров.

Исходя из всего этого, эти приложения востребованы в век технологий и могут получить огромное развитие в дальнейшем. Так как компьютерное зрение очень востребовано в данное время.

Список используемой литературы