МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики –
филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский ядерный университет «МИФИ»
(ИАТЭ НИЯУ МИФИ)

Отделение Интеллектуальные кибернетические системы
Направление подготовки Информационные системы и технологии

Научно-исследовательская работа

Разработка мобильного приложения для преобразования 2D фотографий в 3D вид

Студент группы ИС-Б14 _____ А.В.Кузнецов

Руководитель
к.т.н., доцент отделения ИКС _____ О.А.Мирзеабасов

Обнинск, 2018 г

РЕФЕРАТ

29 стр., 10 рис. , 7 ист.

ANDROID, JAVA, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС, КАРТА ГЛУБИНЫ, АНАЛИЗ, 3D, АНИМАЦИЯ, GIF

Настоящая работа посвящена изучению методов получения трехмерных изображений из двумерных и разработке пользовательского интерфейса мобильного приложения для преобразования 2D фотографий в 3D вид.

Разработанная программа дает возможность преобразовывать простую фотографию в объемную картинку, в виде GIF.

ОПРЕДЕЛЕНИЯ

Референсы — изображения или другие ресурсы для разработки графического интерфейса с учетом существующих, успешно работающих решений

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

UI (user interface) — пользовательский интерфейс

UX (user experience) — пользовательский опыт

ПО —- программное обеспечение

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Основным результатом выполнения проекта будет мобильное приложение, предназначенное для преобразования 2D изображений в 3D вид. Приложение будет распространяться с помощью его размещения в Google Play (для Android-устройств). Соответственно, в качестве основных потребителей создаваемой продукции следует рассматривать владельцев мобильных устройств, которые любят использовать свой телефон или планшет в качестве фотоаппарата. Более того, то подмножество этих пользователей, которые, помимо фотографирования, активно обрабатывают свои фото средствами мобильного устройства и активно делятся этими результатами с друзьями посредством соцсетей.

Поэтому главной целью текущей НИР является изучение разработки мобильного приложения, работы алгоритма преобразования двумерных изображений в трехмерные и разработке современного, функционального и удобного пользовательского интерфейса.

Задачи, решаемые в ходе работы (в соответствии с заданием на НИР):

1) Изучение основ разработки мобильного приложения в среде Android Studio;

2) Основные компоненты пользовательского интерфейса мобильного приложения в Android;

3) Проектирование структуры приложения;

4) Разработка мобильного приложения;

5) Изучение работы алгоритма определения глубины изображения;

6) Подготовка отчета и презентации по НИР.

1. Описание предметной области

В ходе выполнения проекта, главным образом, решается задача преобразования двумерных изображений в трехмерные(GIF) на мобильных устройствах.

В последние годы заметное место в области преобразования и фильтрации изображений занимает задача преобразования двумерных изображений в трехмерные. На сегодняшний день в мире для этого разработаны различные методики, которые позволяют автоматически создавать так называемые «карты глубины» для двумерных изображений, основываясь на свойствах этого изображение и на некоторых предположениях о характере сцены.

1.1. Android Studio

Android Studio - полностью укомплектованная платформа для разработки и тестирования приложений под операционную систему Android. Разработчики этой оболочки (компания Google) внедрили весь необходимый инструментарий для удобного и качественного проектирования новых приложений и доработки существующих. Программа включает в себя такие компоненты как Android SDK, все версии операционки Android, эмулятор для запуска приложений, элементы тестирования и отладки программ.

Создавая новый проект, будет доступна полная структура приложения со всеми файлами, что позволяет более четко и продуманно организовать сам процесс разработки. Очень удобно реализован показ вносимых изменений и дополнений - визуально в реальном времени происходят преобразования в зависимости от заданных действий. Что немаловажно, программа позволяет делать разработку приложений для всех версий ОС Andriod и для различных устройств - можно предварительно оценить внешний вид программы, например, под планшет или смартфон.

Среда Android Studio предназначена как для небольших команд разработчиков мобильных приложений (даже в количестве одного человека), или же крупных международных организаций с GIT или другими подобными системами управления версиями. Опытные разработчики смогут выбрать инструменты, которые больше подходят для масштабных проектов. Решения для

Android разрабатываются в Android Studio с использованием Java или C++. В основе рабочего процесса Android Studio заложен концепт непрерывной интеграции, позволяющий сразу же обнаруживать имеющиеся проблемы. Продолжительная проверка кода обеспечивает возможность эффективной обратной связи с разработчиками. Такая опция позволяет быстрее опубликовать версию мобильного приложения в Google Play App Store. В Android Studio есть удобная маркировка кода, которая позволит без труда ориентироваться в больших проектах. Кроме того, отдельные компоненты можно изменять простым перетаскиванием в другое нужное место, что значительно упрощает редактирование.

1.2. Java

Преимущества языка Java

- Одно из основных преимуществ языка Java — независимость от платформы, на которой выполняются программы: один и тот же код можно запускать под управлением операционных систем Windows, Solaris, Linux, Machintosh и др. Это действительно необходимо, когда программы загружаются через Интернет для последующего выполнения под управлением разных операционных систем.

- Другое преимущество заключается в том, что синтаксис языка Java похож на синтаксис языка C++, и программистам, знающим языки C и C++, его изучение не составляет труда.

- Кроме того, Java — полностью объектно-ориентированный язык, даже в большей степени, чем C++. Все сущности в языке Java являются объектами, за исключением немногих основных типов (primitive types), например чисел.

- Исключена возможность явного выделения и освобождения памяти. Память в языке Java освобождается автоматически с помощью механизма сборки мусора. Программист гарантирован от ошибок, связанных с неправильным использованием памяти.

- Безопасный: методы проверки подлинности основаны на шифровании с открытым ключом.

- Динамический: программирование на Java считается более динамичным, чем на C или C++, так как он предназначен для адаптации к меняющимся условиям. Программы могут выполнять обширное количество во время обработки информации, которая может быть использована для проверки и разрешения доступа к объектам на время выполнения.

1.3. Выбор архитектуры мобильного приложения

В ходе выполнения работы были рассмотрены различные варианты для создания мобильных приложений, предназначенных для преобразования 2D изображений в 3D вид. При этом рассмотрении учитывалось, что результирующие мобильные приложения должны создаваться под операционные системы Android, а также то, что один из основных результатов работы приложения с точки зрения конечного пользователя – это возможность публикации созданного 3D-изображения в виде анимированного gif-файла в одном или нескольких аккаунтов в социальных сетях пользователя. Соответственно, можно исходить из предположения о том, что для функционирования приложения в любом случае необходим доступ к сети интернет. Максимальная унификация различных составных частей приложения между собой хотя бы на уровне исходных кодов вне зависимости от целевой платформы (Android или iOS) является дополнительным преимуществом при рассмотрении различных вариантов создания мобильных приложений.

Один из наиболее простых с технической точки зрения вариантов реализации решения, позволяющего преобразовывать 2D файлы в 3D вид, является решение, основанное на создании веб-сервиса, который предоставляет минимально необходимый пользовательский интерфейс для загрузки желаемого файла на сервер, преобразования файла на сервере и, как результат, возможность скачать получившийся файл на устройство пользователя и поделиться этим файлом в социальных сетях. При простоте архитектуры у этого решения есть один существенных недостаток – как правило, такие решения менее удобны и функциональны, чем нативные (native) мобильные приложения,

разработанные специально под целевую платформу, на которой они будут функционировать.

Рассмотрим два варианта создания нативных мобильных приложений:

1) Использовать наиболее популярные средства разработки и языки программирования для каждой из необходимых мобильных платформ. Создать нативное мобильное приложение, реализующее весь необходимый пользовательский интерфейс, набор сервисных функций. Портировать алгоритм преобразования графического файла из 2D в 3D для локального исполнения на мобильном устройстве. Все необходимые преобразования выполнять локально, на мобильном устройстве. Полученный результат преобразования (анимированный gif) загружать в интернет (социальные сети) по мере его готовности на мобильном устройстве.

2) Использовать наиболее популярные средства разработки и языки программирования для каждой из необходимых мобильных платформ для создания нативных мобильных приложений только для реализации пользовательского интерфейса и набора сервисных функций. Алгоритм преобразования графического файла из 2D в 3D реализуется в виде серверного модуля, соответственно для преобразования выбранного файла и предварительного просмотра полученных результатов необходимо загрузить этот выбранный файл на сервер. Загрузить полученный результат с сервера и поделиться этим результатом в социальных сетях.

Для варианта №1 для операционной системы Android необходимо:

С использованием Android Studio на языке программирования Java реализовать необходимый пользовательский интерфейс, а также весь необходимый набор сервисных функций. Необходимо адаптировать реализацию алгоритма преобразования графического файла из 2D в 3D для использования под управлением операционной системы Android (реализация на C++). Далее, с использованием механизма The Android Native Development Kit (NDK) необходимо обеспечить вызов кода, написанного на языке C++ из «классического» Android-приложения.

Для реализации варианта №2 необходимо:

С использованием Android Studio на языке программирования Java необходимо создать нативное мобильное приложение для реализации пользовательского интерфейса и набора сервисных функций. Эта задача, в целом, является типовой и принципиальных сложностей не вызывает. Алгоритм преобразования графического файла из 2D в 3D следует реализовать в виде серверного модуля, например, для использования под управлением операционной системы Ubuntu. Это обусловлено тем, что Unix-подобные операционные системы имеют существенно более широкое распространение в Web-серверном окружении, чем Windows-сервера.

На основе проведенного исследования моно сделать следующий вывод. С точки зрения скорости, легкости и качества реализации наиболее перспективными являются вариант №2.

Очевидным недостатком подобного решения является существенная его зависимость от скорости и надежности мобильного интернета, а также от доступности конечному пользователю оплаченного траффика. Для обхода этих ограничений предполагается исследовать возможность создания для пользователей ОС Android «самодостаточного» мобильного приложения (вариант №1), которое все необходимые действия, связанные с преобразованием файлов производит непосредственно на мобильном устройстве.

2. Изучение основ разработки мобильного приложения в среде Android Studio

Android основан на Linux. Между приложением и ядром лежит слой API и слой библиотек на нативном коде. Приложение выполняется на виртуальной машине Java (Dalvik Virtual Machine). В Android можно запускать много приложений. Но одно из них есть главным и занимает экран. От текущего приложения можно перейти к предыдущему или запустить новое. Это похоже на браузер с историей просмотров.

Каждый экран пользовательского интерфейса представлен классом Activity в коде. Различные Activity содержатся в процессах. Activity может даже жить дольше процесса. Activity может быть приостановлена и запущена вновь с сохранением всей нужной информации.(рисунок 1)

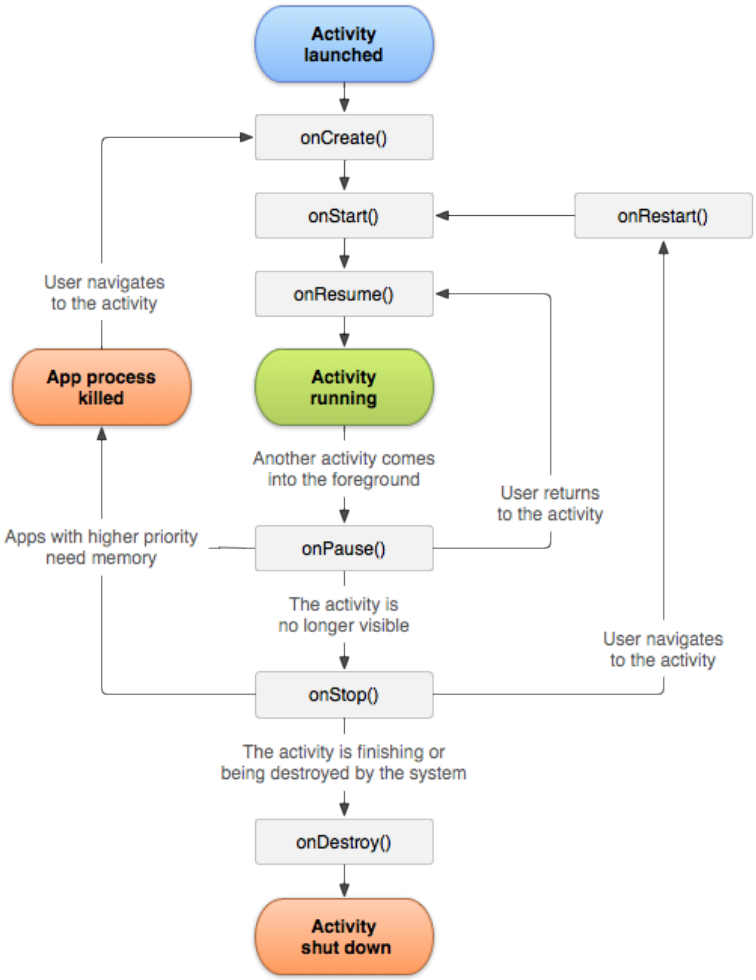

Рисунок 1 – activity

Android использует специальный механизм описания действий основанный на Intent. Когда нужно выполнить действие (сделать звонок, послать письмо, показать окно), вызывается Intent.

Также Android содержит сервисы подобные демонам в Linux для выполнения нужных действий в фоновом режиме (например, проигрывание музыки). Для обмена данными между приложениями используются Content providers (провайдеры содержимого).

Содержимое Activity формируется из различных компонентов, называемых View. Самые распространенные View - это кнопка, поле ввода, чекбокс и т.д. (рисунок 2)

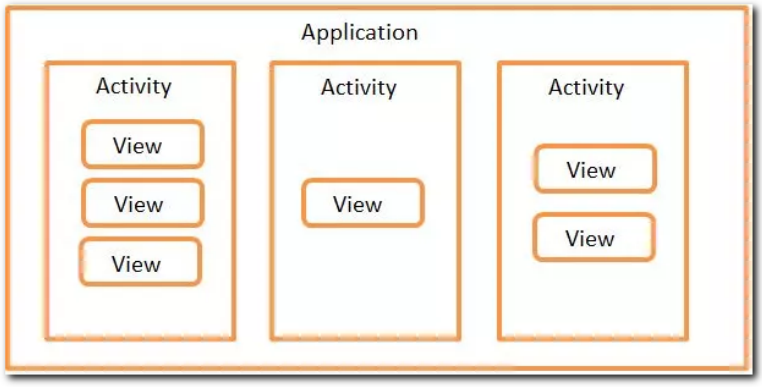

Рисунок 2 – view

Необходимо заметить, что View обычно размещаются в ViewGroup. Самый распространенный пример ViewGroup – это Layout. Layout бывает различных типов и отвечает за то, как будут расположены его дочерние View на экране.

LinearLayout – отображает View-элементы в виде одной строки (если он Horizontal) или одного столбца (если он Vertical).

TableLayout – отображает элементы в виде таблицы, по строкам и столбцам.

RelativeLayout – для каждого элемента настраивается его положение относительно других элементов.

AbsoluteLayout – для каждого элемента указывается явная позиция на экране в системе координат (x,y)

3. Основные компоненты пользовательского интерфейса мобильного приложения в Android

В Android используется UI-фреймворк, сравнимый с другими полнофункциональными UI-фреймворками, применяемыми на локальных компьютерах. Он является более современным и асинхронным по природе. По существу, UI-фреймворк Android относится уже к четвертому поколению, если считать первым поколением традиционный прикладной интерфейс программирования Microsoft Windows, основанный на C, а MFC (Microsoft Foundation Classes, библиотека базовых классов Microsoft на основе C++) - вторым. В таком случае UI-фреймворк Swing, основанный на Java, будет третьим поколением, так как предлагаемые в нем возможности дизайна значительно пре-

восходят по гибкости MFC. Android UI, JavaFX, Microsoft Silverlight и язык пользовательских интерфейсов Mozilla XML (XUL) относятся к новому типу UI-фреймворков четвертого поколения, в котором UI является декларативным и поддерживает независимую темизацию.

При программировании в пользовательском интерфейсе Android применяется объявление интерфейса в файлах XML. Затем эти определения представления (view definitions) XML загружаются в приложение с пользовательским интерфейсом как окна. Даже меню приложения загружаются из файлов XML. Экраны (окна) Android часто называются активностями (activities), которые включают в себя несколько видов, нужных пользователю, чтобы выполнить логический элемент процесса. Виды (views) являются основными элементами, из которых в Android состоит пользовательский интерфейс. Виды можно объединять в группы (view groups). Для внутренней организации видов используются давно известные в программировании концепции холст (canvas), рисование (painting) и взаимодействие пользователя с системой (user interaction).

Такие составные представления, в которые входят виды и группы видов, работают на базе специального логического заменяемого компонента пользовательского интерфейса Android.

Одной из ключевых концепций фреймворка Android является управление жизненным циклом (lifecycle) окон явлений (activity windows). В системе применяются протоколы, поэтому Android может управлять ситуацией по мере того, как пользователи скрывают, восстанавливают, останавливают и закрывают окна явлений.

4. Проектирование структуры приложения

Одна из важнейших задач в ходе создания мобильного приложения, преобразовывающего 2D снимки в объемные (3D) - разработка удобного и интуитивно-понятного пользовательского интерфейса. UI/UX (user Interface, user experience) составляющая, она же пользовательский интерфейс и пользовательский опыт, является более чем просто значимым элементом современного мобильного приложения. Удобство расположения элементов управления и приятное ви-

зуальное оформление напрямую влияют на настроение пользователя при использовании продукты. Именно некачественный UI/UX-дизайн отпугивает людей от использования многих приложений в пользу их более достойных альтернатив.

На главный экран камеры (рисунок 3) выведены следующие функции:

- Спуск затвора;
- Выбор фото из галереи;
- Смена камеры;
- Управление вспышкой;
- Переход к настройкам и справке.

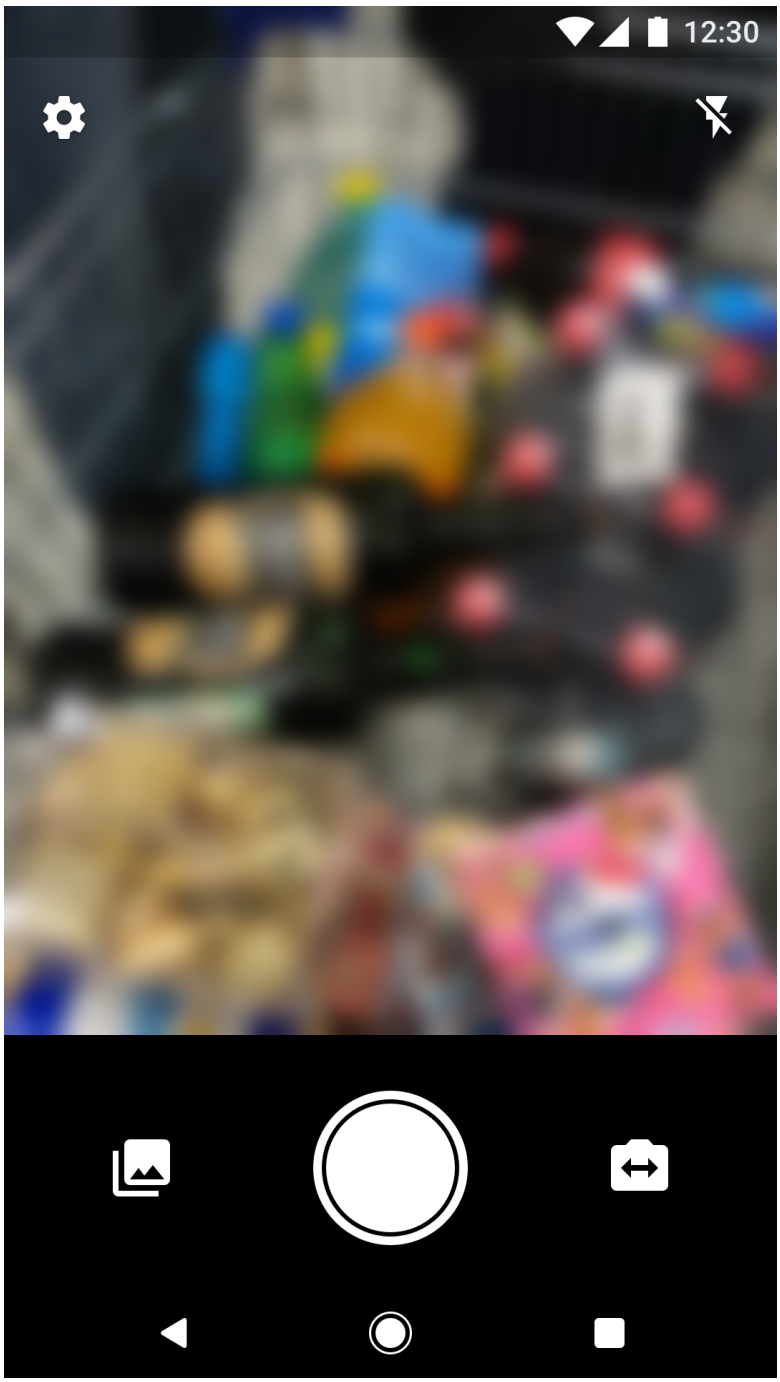

Рисунок 3 – Окно с камерой

На экране с обработанной фотографией по нажатии кнопки «Далее» появляется bottom sheet (рисунок 4), включающий в себя быстрые функции шеринга и сохранения полученного фото в галерею.

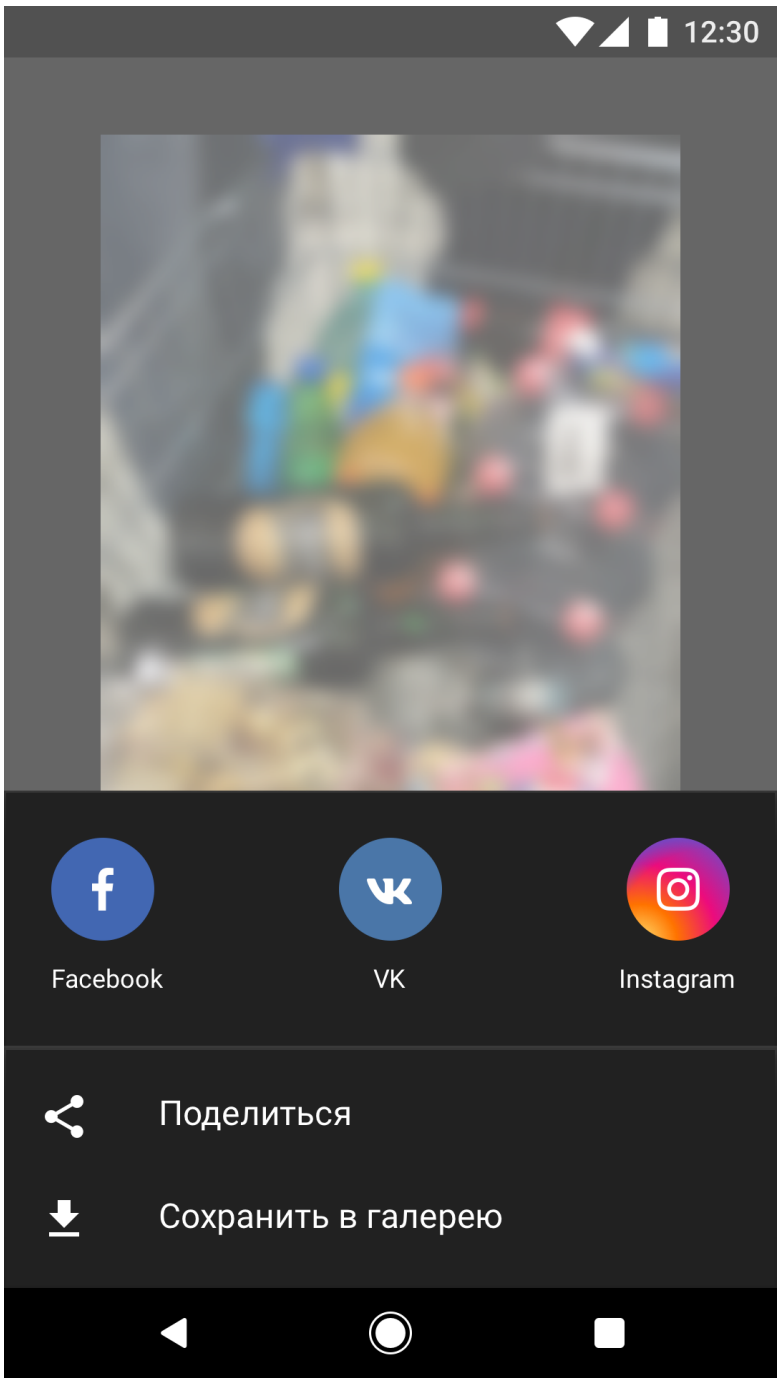

Рисунок 4 – Окно с просмотром фото

При создании пользовательского интерфейса приложения были проанализированы современные, с аналогичным функционалом мобильные приложения в целом. Был сделан акцент на необходимости создания современного, функционального и не перегруженного пользовательского интерфейса. В результате проведенного анализа был создан пользовательский интерфейс мобильного приложения для преобразования 2D фотографий в 3D вид.

5. Разработка мобильного приложения

Рассмотрю практичный пример, когда программно запускаю приложение "Камера а полученную фотографию сохраняю в папке. [**camera**]

В манифесте нужно добавить разрешение на запись файла в хранилище и указать требование наличия камеры.

Используем статическую константу ACTION_IMAGE_CAPTURE из объекта MediaStore для создания намерения, которое потом нужно передать методу startActivityForResult(). Разместим на форме кнопку и ImageView, в который будем помещать полученный снимок. Полученное с камеры изображение можно обработать в методе onActivityResult()

При тестировании примера на своём телефоне я обнаружил небольшую проблему - когда снимок передавался обратно на моё приложение, то оно находилось в альбомном режиме, а потом возвращалось в портретный режим. При этом полученный снимок терялся. Поэтому перед нажатием кнопки я поворачивал телефон в альбомный режим, чтобы пример работал корректно. Поэтому надо предусмотреть подобное поведение, например, запретить приложению реагировать на поворот и таким образом избежать перезапуска Activity.

По умолчанию фотография возвращается в виде объекта Bitmap, содержащего миниатюру. Этот объект находится в параметре data, передаваемом в метод onActivityResult(). Чтобы получить миниатюру в виде объекта Bitmap, нужно вызвать метод getParcelableExtra() из намерения, передав ему строковое значение data.

(рисунок 5)

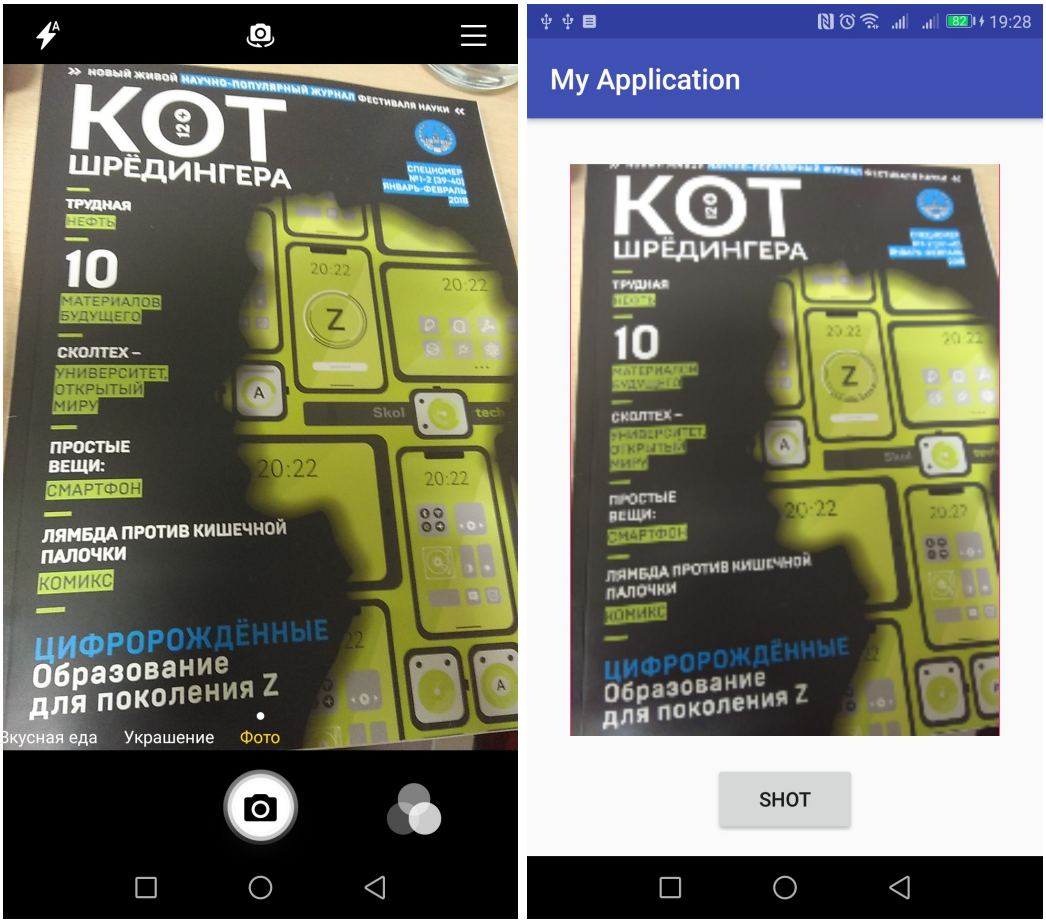

Рисунок 5 – Результат работы приложения

6. Восстановление глубины из одного расфокусированного изображения

Рассмотрим сложную задачу восстановления глубины из одного расфокусированного изображения. Входное расфокусированное изображение повторно размыто с использованием гауссова ядра, а значение размытости размытия может быть получено из коэффициента градиента между входными и повторно размытыми изображениями. Распространяя количество размытия в крайних положениях на все изображение, можно восстановить всю карту глубины сцены.

Результат восстановления глубины нашего метода. (рисунок 6) Большая интенсивность означает большую глубину на всех картах глубины

Рисунок 6 – Входное изображение и карта глубины

Сосредоточимся на более сложной проблеме восстановления относительной глубины из одного расфокусированного изображения, захваченного некалиброванной обычной камерой. Метод обратной диффузии [**Proc**] моделирует размытие дефокусировки в качестве процесса диффузии тепла и использует неоднородную диффузию тепла для оценки размытости размытия в краевых положениях. В отличие от этого, моделируем размытие дефокусировки как размытие 2D Gaussian. Входное изображение повторно размывается с использованием известного гауссовского размытия, и рассчитывается коэффициент градиента между входными и повторно размытыми изображениями. Величина размытия в краевых местоположениях может быть получена из отношения.

Рассмотрим эффективный метод оценки размытия, основанный на гаус-

совском градиентном соотношении, и показываем, что он устойчив к шуму, неточному расположению краев и помехам от соседних ребер. Без каких-либо изменений в камерах или при использовании дополнительного освещения наш метод позволяет получить карту глубины сцены, используя только одно расфокусированное изображение, снятое обычной камерой. Как показано (рисунок 6), этот метод может извлекать карту глубины сцены с довольно высокой степенью точности.

6.1. Модель дефокусировки

Оцениваем размытие размытия в местах краев и предполагаем, что края являются ступенчатыми краями. Идеальный край шага может быть смоделирован как:

$$f(x) = Au(x) + B \tag{1}$$

где $u(x)$ - ступенчатая функция. A и B - амплитуда и смещение края соответственно. Обратите внимание, что ребро расположено в точке $x = 0$.

Предположим, что фокус и дефокусировка подчиняются тонкой модели объектива [**Optics**]. Когда объект размещается на расстоянии фокусировки d_f, все лучи от точки объекта будут сходиться к одной точке датчика, и изображение будет резким. Лучи из точки другого объекта на расстоянии d достигают нескольких точек датчика и приводят к размытому изображению. Размытый рисунок зависит от формы апертуры и называется кругом путаницы (CoC) [**Optics**]. Диаметр CoC характеризует величину расфокусировки и может быть записан как

$$c = \frac{|d - d_f|}{d} \frac{f_0^2}{N(d_f - f_0)} \tag{2}$$

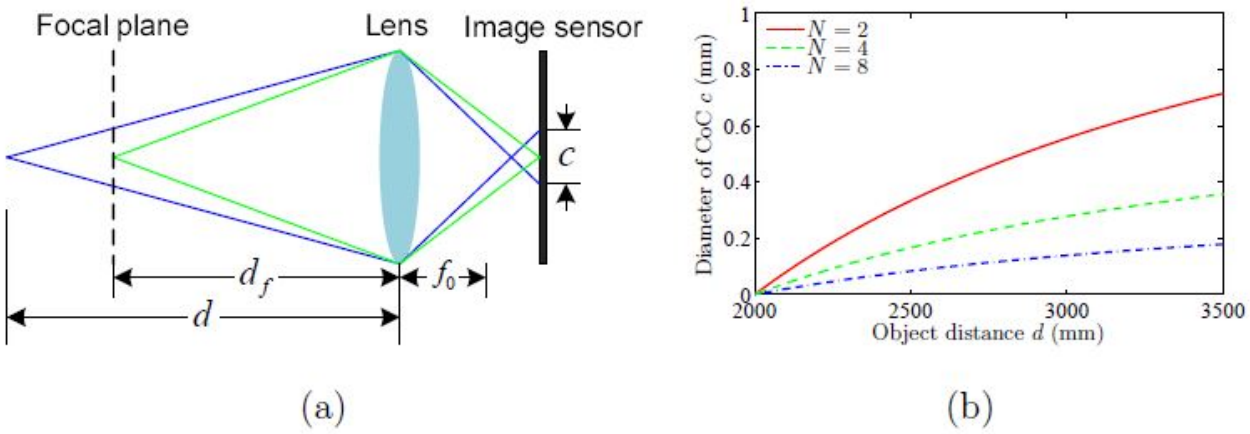

Рисунок 7 – Тонкая модель объектива

где f_0 и N - фокусное расстояние и номер остановки камеры соответственно. На рисунке 7 показаны фокус и расфокусировка для модели тонких линз и как изменяется диаметр круга замешательства с d и N, при фиксированном f_0 и d_f. Как мы видим, диаметр CoC c является нелинейной монотонно возрастающей функцией расстояния объекта d. Размытие дефокусировки может быть смоделировано как свертка острого изображения с функцией распределения точек (PSF). PSF обычно аппроксимируется гауссовой функцией $g(x, \sigma)$, где стандартное отклонение $\sigma = kc$ пропорционально диаметру CoC c. Используем σ как меру глубины сцены. Следовательно, размытие ребра $i(x)$ можно представить следующим образом,

$$i(x) = f(x) \otimes g(x, \sigma) \tag{3}$$

6.2. Оценка размытости

На рисунке 8 показан обзор метода оценки размытия. Граница шага повторно размыта с использованием гауссова ядра с известным стандартным отклонением. затем рассчитывается соотношение между величиной градиента края ступени и ее размытой версией. Это максимальное значение в краевом положении. Используя максимальное значение, можем вычислить величину размытости дефокусировки в местоположении края.

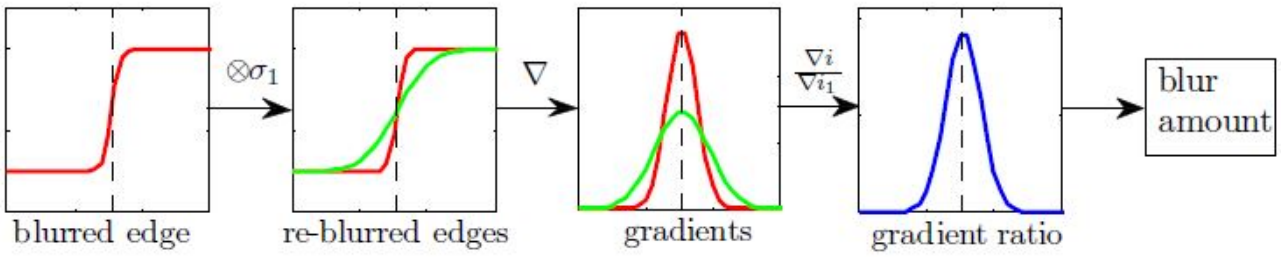

Рисунок 8 – Оценка размытости, черная штриховая линия обозначает местоположение края

Используем двумерное изотропное гауссовское ядро для повторного размытия и величину градиента можно вычислить следующим образом:

$$|| \bigtriangledown i(x,y)|| = \sqrt{\bigtriangledown i_x^2 + \bigtriangledown i_y^2} \tag{4}$$

где $\bigtriangledown i_x$ и $\bigtriangledown i_y$- градиенты вдоль направлений x и y соответственно. Устанавливаем повторное размытие $\sigma_0 = 1$ и используем детектор края Canny [**IEEE**] для выполнения обнаружения края. Шкала размытия оценивается в каждом краевом положении, образуя редкую карта глубины, обозначаемая $\hat{d}(x)$. Однако из-за шума или мягких теней оценки размытия могут содержать некоторые ошибки. Чтобы подавить эти ошибки, применяется совместный двусторонний фильтр [**ACM**] на разреженной карте глубин $\hat{d}(x)$. Выход совместного двустороннего фильтра в каждом краевом положении x определяется как:

$$BF(\hat{d}(x)) = \frac{1}{W(x)} \sum_{y \in N(x)} G_{\sigma s}(||x-y||) G_{\sigma r}(||I(x)-I(j)||) \hat{d}(y) \tag{5}$$

где $W(x)$ - коэффициент нормировки, а $N(x)$ - окрестность точки x, заданная размером пространственного гауссовского фильтра $G_{\sigma s}$. Фильтрация выполняется только по краям. Совместный двусторонний фильтр корректирует некоторые ошибки в разреженной карте глубины и избегать распространения этих ошибок в интерполяции глубины, описанной в следующем разделе.

6.3. Интерполяция глубины

Данный метод оценки размытия дает разреженную карту глубин $d(x)$ с оценками глубины в местах краев. Чтобы получить полную карту глубины

$d(x)$, необходимо распространить эти значения из местоположений краев на все изображение. Найдем регуляризованное отображение глубины $d(x)$, которое близко к разреженной карте глубин $d(x)$ в каждом краевом положении. Для этих задач обычно используются методы интерполяции с учетом края [**ACM2**]. Здесь применяется матирующий лапласиан [**IEEE2**] для выполнения интерполяции. Мы переписываем разреженную карту глубин d (x) и полное отображение глубины d (x) в их векторной форме как d и d. Тогда проблему интерполяции глубины можно сформулировать как минимизирующую следующую функцию стоимости:

6.4. Эксперименты

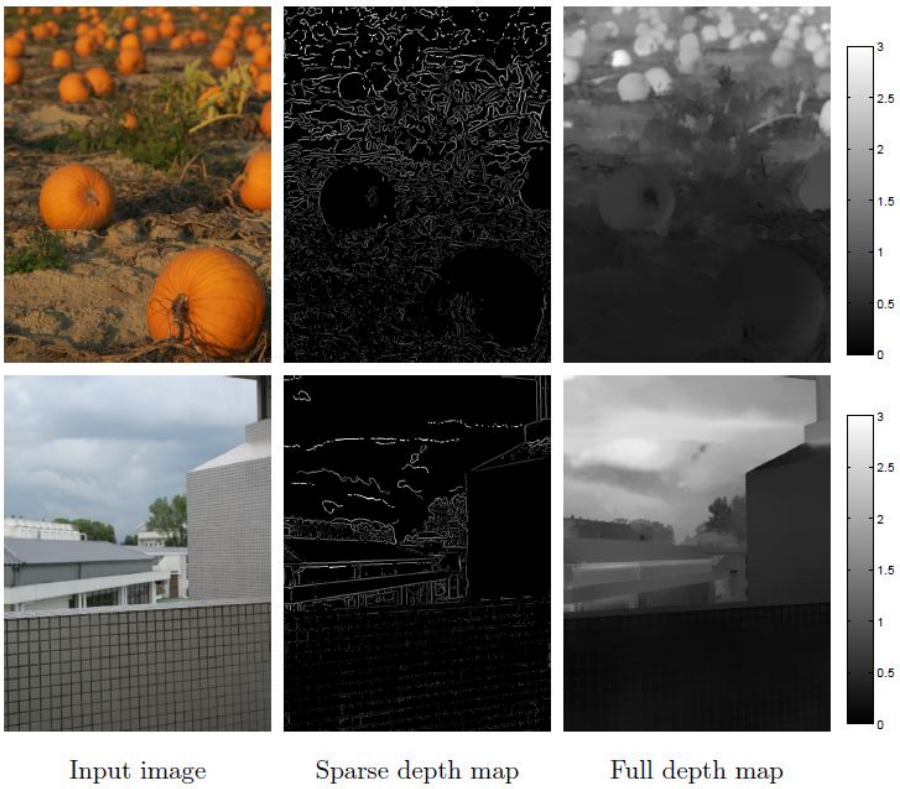

Рисунок 9 – Восстановление глубины на реальных изображениях. Наш метод может работать как на сценах с непрерывной глубиной (изображение тыквы), так и на слоистой глубине (изображение здания) для получения карты глубины с довольно хорошей точностью.

Как показано на рисунке 9, я тестирую наш метод на некоторых реальных изображениях. В изображении тыквы глубина сцены непрерывно изменяется

от нижней к верхней части изображения. Оценочная карта глубины фиксирует непрерывное изменение глубины. В изображении здания сцена в основном содержит три слоя: стены, дом и небо. Наш метод позволяет создавать карты глубин, точно представляющие эти слои глубины. Как видно из результатов, наш метод позволяет точно восстановить глубину сцены из одного расфокусированного изображения. На рисунке 10 мы сравниваем наш метод с методом обратной диффузии [**Proc**].

Рисунок 10 – Сравнение нашего метода с методом обратной диффузии, на примере цветка

Метод обратной диффузии создает грубую слоистую карту глубины. В результате этого цветочный слой плохо отделен фоновыми слоями и содержит некоторые оценки погрешности. Напротив, наш метод способен производить более точную и непрерывную карту глубины, а цветочный слой хорошо отделен фоновой травой и деревьями.

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы изучил основы разработки мобильного приложения в среде Android Studio. Также была изучена работа алгоритма определения глубины изображения.

При создании пользовательского интерфейса приложения был сделан акцент на необходимости создания современного, функционального и не перегруженного пользовательского интерфейса. В результате было создано мобильное приложение, с помощью которого можно сделать снимок и сохранить его в галерею.

Приложение А

Листинг 1 – Часть кода реализации класса `MainActivity`

```
1  public class MainActivity extends AppCompatActivity {
2          ImageView ivPreview;
3          String mCurrentPhotoPath;
4          @Override
5          protected void onCreate(Bundle savedInstanceState) {
6                  super.onCreate(savedInstanceState);
7                  setContentView(R.layout.activity_main);
8                  ivPreview = findViewById(R.id.image);
9                  Button btn = findViewById(R.id.button);
10                 btn.setOnClickListener(new View.OnClickListener() {
11                         @Override
12                         public void onClick(View v) {
13                                 dispatchTakePictureIntent();
14                         }
15                 });
16         }

17         private void dispatchTakePictureIntent() {
18                 Intent pictureIntent = new Intent(
19                 MediaStore.ACTION_IMAGE_CAPTURE);
20                 Intent pictureIntent2 = new Intent(
21                 MediaStore.ACTION_IMAGE_CAPTURE);
22                 if(pictureIntent.resolveActivity(getPackageManager())
23                         //Create a file to store the image
24                         File photoFile = null;
25                         try {
26                                 photoFile = createImageFile();
27                         } catch (IOException ex) {
```

```
28                              // Error occurred while creating the F
29                      }
30                      if (photoFile != null) {
31                              Uri photoURI = FileProvider.getUriForF
32                              Intent mediaScanIntent = new Intent(In
33                              File f = new File(mCurrentPhotoPath);
34                              mediaScanIntent.setData(photoURI);
35                              this.sendBroadcast(mediaScanIntent);
36                              pictureIntent.putExtra(MediaStore.EXTR
37                              photoURI);
38                              startActivityForResult(pictureIntent2,
39                              1);
40                      }
41              }
42      }

43      private File createImageFile() throws IOException {
44              // Create an image file name
45              String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmm
46              String imageFileName = "JPEG_" + timeStamp + "_";
47              File storageDir = getExternalFilesDir(Environment.DIRE
48              File image = File.createTempFile(
49              imageFileName,  /* prefix */
50              ".jpg",         /* suffix */
51              storageDir      /* directory */
52              );

53              // Save a file: path for use with ACTION_VIEW intents
54              mCurrentPhotoPath = image.getAbsolutePath();
55              return image;
56      }

57      @Override
```

```
58          protected void onActivityResult(int requestCode, int resultCod
59                  if (requestCode == 1 && resultCode == RESULT_OK) {
60                          Bitmap imageBitmap = (Bitmap) data.getExtras()
61                          ivPreview.setImageBitmap(imageBitmap);
62                  }
63          }
64  }
```
