Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

РЕАЛИЗАЦИЯ МНОГОФУНКЦИОНАЛЬНОГО ПОМОЩНИКА С ПРИМЕНЕНИЕМ АЛГОРИТМОВ МАШИННОГО ЗРЕНИЯ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Казеевой Асель Александровны

Научный руководитель
доцент, к. ф.-м. н., доцент _____ А. С. Иванов

Заведующий кафедрой
к.ф.-м.н. _____ С. В. Миронов

Саратов 2016

ВВЕДЕНИЕ

Компьютерное зрение — перспективная область, зародившаяся как часть анализа искусственного интеллекта, но получившая в дальнейшем распространение в самых различных сферах: расчет траектории для пути робота, считывание штрихкодов, дополненная реальность. В то же время совершенствуются технологии, позволяющие распознавать естественную речь и выполнять запросы пользователя по голосовым командам; данные технологии получили развитие в цифровых помощниках, таких как $Siri$ (сервис планирования для смартфонов), $Rearden\ Personal\ Assistant$ (веб-сервис для бизнес-планирования). Закономерно использование алгоритмов компьютерного зрения для расширения функциональности цифровых помощников и для улучшения их интерфейсов.

Целью данной бакалаврской работы является исследование методов компьютерного зрения, средств языка $Python$ и реализация многофункционального виртуального помощника с элементами дополненной реальности.

Дипломная работа состоит из введения, трех разделов, заключения, списка использованных источников и двух приложений.

В разделе 1 приводится определение компьютерного зрения, основные этапы развития этой области. Также описывается проективная модель камеры, внутренние параметры камеры, эффект дисторсии.

Во 2 разделе приводятся основные понятия, связанные с алгоритмами компьютерного зрения, далее описываются алгоритмы, использованные в программной реализации: детектор Канни, применяемый для определения маркера, алгоритм для распознавания жестов, основанный на дефектах выпуклости фигур. Затем описывается библиотека OpenCV, реализующая основные метода и алгоритмы компьютерного зрения и используемая в созданной программе.

В 3 разделе описывается назначение, интерфейс и детали реализации многофункционального виртуального помощника.

В приложении А содержится код файлов программы «Виртуальный ассистент» `menu.py` и `translator.py`, в приложении Б — CD-диск с презентацией для защиты бакалаврской работы и реализацией программы «Виртуальный помощник».

1 Основное содержание работы

За время своей жизни мозг человека анализирует огромное количество различной информации, совершенствуется на основе полученного опыта и в числе прочих задач учится распознавать сложные объекты.

Для того чтобы добиться похожей эффективности от программы, исследователи компьютерного зрения развивают математические методы для воссоздания трехмерных форм и внешнего вида различных объектов на изображении. На данный момент существуют методы для вычисления трехмерной модели области на основании множества частично перекрывающихся фотографий, отслеживания движения человека на неоднородном фоне, поиск людей по комбинации характеристик (лицо, одежда, волосы).

По определению цель компьютерного зрения заключается в формировании полезных выводов относительно объектов и сцен реального мира на основе анализа изображений, полученных с помощью датчиков. [1]

Теория компьютерного зрения за время своего развития прошла через несколько этапов: от определения основных понятий и задач, решаемых системами компьютерного зрения, информационного подхода Дэвида Марра до применения методов машинного обучения для повышения эффективности алгоритмов. [2]

Также, поскольку источниками изображений для анализа является фотоаппаратура, для реализации алгоритмов компьютерного зрения надо сначала исследовать ее работу.

Работу современных камер можно описать с помощью модели точечной перспективы, которая была создана еще в 15 веке. Так как в процессе перспективной проекции формируется перевернутое изображение, вместо него возможно анализировать мнимое изображение, расположенное перед отверстием камеры на равном расстоянии с плоскостью изображения [3]

При использовании видео и фотооборудования может возникнуть эффект дисторсии: аберрации, вызывающей искажение изображения прямых линий, в результате чего нарушается подобие между объектом и его изображением.

Для устранение дисторсии координаты пикселей можно пересчитать с помощью следующего уравнения:

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 xy + p_2(r^2 + 2x^2)$$
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_2 xy + p_1(r^2 + 2y^2),$$

где (x, y) — первоначальное расположение пикселя, $(x_{corrected}, y_{corrected})$ — расположение пикселя после устранения геометрических искажений, $k1,\ k2,\ k3$ — коэффициенты радиальной дисторсии, $p1,\ p2$ — коэффициенты тангенциальной дисторсии, $r^2 = x^2 + y^2$ [4].

Для вычисления внутренних и внешних параметров камеры, надо использовать набор изображений с хорошо определяемым графическим паттерном, например, шахматной доской. Далее распознаются особые точки (в случае шахматной доски — точки на пересечении черных и белых квадратов), а после, зная координаты в реальном мире и изображении вычисляются коэффициенты дисторсии. Для улучшения результата обычно применяется набор из нескольких изображений. [5]

Для дальнейшего исследования видеопотока или улучшения качества изображения в алгоритмах компьютерного зрения применяется предварительная обработка: фильтрация, поиск и выделение краев, морфологические операции.

Одним из базовых понятий в компьютерном зрении является свертка — математическая операция, применяемая к функциям f и g и дающая в результате третью функцию: в обработке изображений свертка используется для вычисления значения текущего пикселя по значениям соседних пикселей путем перемножения элементов матриц. Один из массивов обычно представлен серым изображением, другой (также двумерный) образует ядро свертки.

Свертка выполняется путем перемещения ядра по изображению, как правило, начиная с верхнего левого угла. Каждая позиция ядра определяет значение выходного пикселя, которое вычисляется перемножением значений пикселя ядра и соответствующего значения пикселя изображения и сложением полученных значений. [6]

Для шумоподавления на стадиях предварительной обработки в алгоритмах компьютерного зрения применяется фильтр размытия по Гауссу: название фильтра произошло от нормального или Гауссовского распределения, используемого для вычисления значений пикселя. [7]

Другая важная задача в алгоритмах компьютерного зрения — проблема определения границ.

Джон Канни тщательно исследовал эту проблему и в итоге сформулировал ряд критериев для определения наилучшего способа решения задачи:

— Алгоритм должен определять каждую границу не более одного раза (таким образом, широкая зона колебаний яркости будет обрабатываться как одна граница)
— Не реагировать на ложные случаи, по возможности отфильтровывать края, порожденные шумом
— Точно локализовать границу, без фрагментации [8] [9]

Следующий метод, рассмотренный в бакалаврской работе — реализация дополненной реальности.

Дополненную реальность (также называемую смешанной реальностью) можно рассматривать как систему, объединяющую в себе виртуальные и реальные предметы, действующую в реальном времени, находящуюся в трехмерном пространстве.

В основе построения систем дополненной реальности лежат технологии двух типов в зависимости от применения маркера. Методы, не использующие маркеры, нередко используются в смартфонах и используют данные, полученные от GPS-навигатора.

Маркер — это некоторый объект, обладающий особыми визуальными характеристиками, который может быть распознан программой для воссоздания виртуального объекта: трехмерной фигуры, анимации, карты, инфографики, справочной информации. Основную сложность в отслеживании маркера представляет анализ его точного расположения, дополнительно применяются последние достижения компьютерной графики для создания иллюзии присутствия объекта в окружающем пространстве. [10]

Часто в качестве маркера применяют лист с неким хорошо различимым паттерном, однако точный вид маркера зависит от применяемых для его поиска методик. Внешний вид маркера не ограничивается бумагой, маркер может быть представлен предметами с простым контуром, ладонью человека.

В ходе выполнения бакалаврской работы была реализована программа «Virtual assistant», использующая вышеописанные методы.

Исходный код программы состоит из пятнадцати файлов с расширением $.py$. Файл $main.py$ определяет главный класс, с него начинается запуск программы: внутри описан класс $VirtualAssistant$, отвечающий за инициализацию OpenGL и служебных классов. В файле $markers.py$ описана функция, выполняющая поэтапный поиск маркера на кадре; дополнительные функции,

участвующие в распознавании маркера, расположены в $markersfunction.py$. В $camera.py$ создается поток для управления веб-камерой.

В файлах $mp3player$, $news$, $translator$ и $wiki$ реализованы функции музыкального плейера, поиска новостей, переводчика распознанного текста и поиска по Википедии соответственно.

Файлы $calibration.py$ и $createcabfile.py$ отвечают за калибровку.

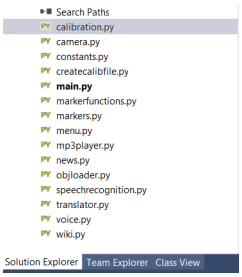

Рисунок 1 – Структура программы

Программа имеет голосовой интерфейс, реагирует на установленные команды и может отвечать на английском языке.

При запуске появляются два окна: окно веб-камеры и консольное окно, выдающее текстовую информацию, возникающую в процессе выполнения программы.

В случае обнаружения маркера программа формирует трехмерный объект (в данном случае - робота, рисунок 2), после чего возможно отдать голосовую команду для запуска меню. Для распознания голоса использовалась библиотека $speachrecognition$, для преобразования текста в голос $pyttsx$.

Рисунок 2 – Фигура робота, изображенная поверх маркера

По команде «menu» запускается окно, созданное средствами OpenCV, разделенное на четыре части, представляющие различные функции команды. В левом верхнем углу расположена иконка с нотой, соответствующая модулю

музыкального плеера, в левом верхнем — иконка с изображением газеты, запускающая модуль новостей, в нижнем левом — изображение книги, определяющее функцию переводчика, в правом нижнем — символ Википедии, связанный с модулем поиска по энциклопедии (рисунок 3).

Иконки отрисовываются поверх видео-потока, чтобы пользователь смог определить движение руки относительно камеры и понять, правильно ли указывает ладонью на иконку. Для наглядности также выводится изображение контуров обнаруженных предметов. Для определения ладони был использован алгоритм определения дефектов выпуклости.

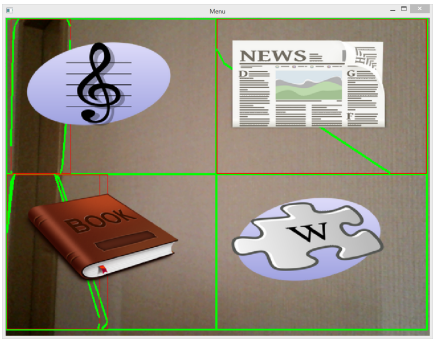

Рисунок 3 – Окно меню

После обнаружения ладони в заданном интервале запускается соответствующая функция, окно меню закрывается для сокращения вычислительной нагрузки. В один момент может быть запущена одна функция.

Голосовые команды обрабатываются в потоке главного модуля, поэтому после получения команды «Close» модуль работающей функции закрывается, и модуль меню повторно запускается.

Одна из реализованных функций виртуального помощника — перевод текста с изображения веб-камеры. Для распознавания текста использовался модуль $PyTesser$, для перевода библиотека $translate$, использующая для перевода
$http://mymemory.translated.net/$ с установленным ограничением 1000 слов в день. После запуска модуля открывается окно OpenCV, разделенное на две части: справа выводится информация с короткой инструкцией для использования модуля, слева изображение с веб-камеры.

К окну слева необходимо поднести текст на английском языке, как только границы листа совпали с границами выделенной области, отдается команда. По голосовой команде «Start» делается снимок, изображение обрабатывает-

ся, распознается текст и выполняется перевод полученного отрывка (рисунок 4).Текст перевода выводится в окне справа, где ранее были инструкции, а распознанный английский текст выводится в окне интерпретатора. Если перевод был выполнен корректно и пользователь хочет сохранить результат, отдается голосовая команда «Save». Результат сохраняется в файл `translation.txt`.

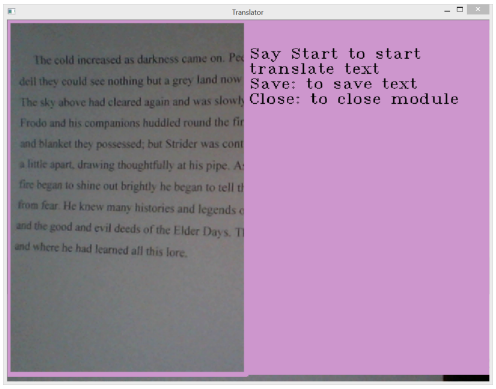

Рисунок 4 – Окно переводчика

Для получения видео-потока использовалась веб-камера ноутбука с разрешением 1.3 Мп. В реализации функции для улучшения качества изображения применялось нерезкое маскирование, однако полученное с веб-камеры нечеткое изображение плохо поддавалось распознаванию текста и в результате не удалось добиться приемлемого уровня машинного перевода (рисунок 5).

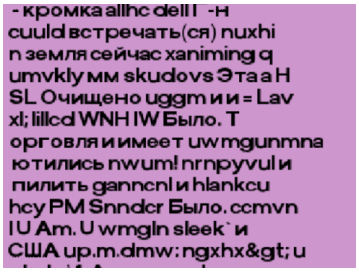

Рисунок 5 – Пример переведенного текста

Функция была проверена для загружаемого изображения текста высокого качества, определение текста и его перевод были выполнены с хорошим результатом. Следовательно, для улучшения работы приложения необходимо использовать камеру с большим разрешением.

Также виртуальный помощник может искать новости с помощью библиотеки $newspaper$ для $Python$: данная библиотека позволяет получать новости на множестве языков с различных сайтов в удобном формате.

Источниками информации являются объекты (source objects), абстрагирующие данные с популярных новостных ресурсов (например, CNN или ESPN). Построение объекта извлечет описание сайта и информацию о новостных лентах, статьях, категориях, содержащихся на нем.

Главный недостаток в том, что загружаемые объекты довольно объемны и, если они загружаются в процессе работы, программа перестает корректно работать. Другой выход — инициализировать объект при запуске программы, что замедляет загрузку. Так, окно виртуального помощника с веб-камерой запускается на несколько минут позже окна с консолью $Anaconda$.

Построение объекта `self.paper` при инициализации объекта класса $News$ ($http://espn.com$ — это сайт, посвященный спортивной тематике):

```
import newspaper

 def __init__(self):
          self.voice = VoiceOfAssistant()
          self.understanding = UnderstandingVoice()
          self.is_enabled = False
          self.voice.start()
          self.understanding.start()
          self.articles = np.array([None, None, None])
          self.request = None
          self.paper = newspaper.build('http://espn.com',
                  memoize_articles=False, language = 'en')

```

Другая функция виртуального помощника реализует интуитивное управление музыкальным плейером. После запуска функции открывается окно OpenCV, разделенное на три горизонтальные части, в каждой части поверх изображения с веб-камеры написано название песни. Песни расположены в папке проекта в директории `\Music`. Названия песен перечисляются по алфавитному порядку, в один момент отображается три названия (рисунок 6).

По голосовой команде «Next» загружаются следующие три песни, по команде «Back» предыдущие три песни. Если это невозможно (например, все песни загружены), выводится сообщение в командную строку.

По движению руки запускается песня (рисунок **??**). Голосовыми командами «Stop» и «Play» выполняется остановка и запуск проигрываемой музы-

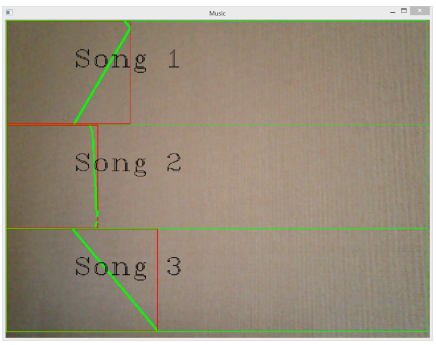

Рисунок 6 – Окно музыкального плейера

кальной композиции (для успешного распознания команд и снижения уровня шум в комнате песню стоит прослушивать в наушниках).

Последняя из функции виртуального помощника — поиск информации в Википедии. Запуск функции открывает окно OpenCV, представленное однотонным четырехугольником со служебной информацией.

В модуле функции была использована библиотека *wikipedia*, реализующая API для доступа к данным Википедии. Библиотека позволяет выполнять поиск по статьям, загружать изображения, получать синопсис каждой статьи. По команде «read N», где N — искомая тема, выполняется запрос. Результат со списком найденных статей, содержащих N, выводится в окне: рисунок 7

Рисунок 7 – Список найденных статей

Получив список статей, пользователь может воспользоваться второй голосовой командой для загрузки текста статьи: по команде «read N» в окно модуля выводятся первые три предложения краткого содержания статьи. По команде «Save» статья загружается в текст с названием, содержащем заголовок статьи `N.txt`.

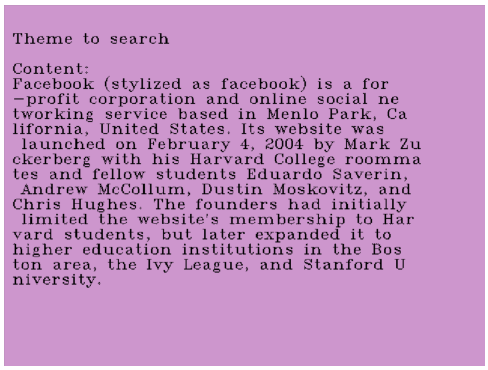

Рисунок 8 – Текст статьи

ЗАКЛЮЧЕНИЕ

Основной задачей дипломной работы было исследование компьютерного зрения и его алгоритмов, реализация программы виртуального помощника. В результате на языке $Python$ была реализована программа, использующая обнаружение маркера для создания объекта дополненной реальности. Созданная программа интерактивна, отвечает на движение рук и голосовые команды, которые запускают выполнение одной из следующих функций:

- музыкальный плейер
- получение новостей с заданного в программе ресурса
- перевод текста, полученного с веб-камеры
- чтение статей с Wikipedia, сохранение полученной информации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Шапиро, Л.* Компьютерное зрение / Л. Шапиро, Д. Стокман. — Москва: БИНОМ. Лаборатория знаний, 2006.

2 Системы компьютерного зрения: современные задачи и методы [Электронный ресурс]. — URL: http://www.controlengrussia.com/innovatsii/sistemy-komp-yuternogo-zreniya-sovremenny-e-zadachi-metody/ (Дата обращения 15.05.2016). Загл. с экр. Яз. рус.

3 *Форсайт, Д.* Компьютерное зрение: современный подход / Д. Форсайт, Ж. Понс. — Москва: Издательский дом «Вильямс», 2004.

4 Калибровка Kinect v2 с помощью OpenCV на Python[Электронный ресурс]. — URL: https://habrahabr.ru/post/272629/ (Дата обращения 10.05.2016). Загл. с экр. Яз. рус.

5 Camera Calibration [Электронный ресурс]. — URL: http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html (Дата обращения 12.05.2016). Загл. с экр. Яз. англ.

6 *Howse, J.* OpenCV Computer Vision with Python / J. Howse. — Birmingham: Packt Publishing, 2013.

7 Matching found glyph with database of glyphs[Электронный ресурс]. — URL: http://www.aforgenet.com/articles/glyph_recognition/ (Дата обращения 17.05.2016). Загл. с экр. Яз. англ.

8 *Szeliski, R.* Computer vision: algorithms and applications / R. Szeliski. — London: Springer, 2011.

9 *Bradski, G.* Learning OpenCV. Computer Vision with the OpenCV Library / G. Bradski, A. Kaehler. — Sebastopol: O'Reilly Media, 2008.

10 Технологии и алгоритмы дополненной реальности[Электронный ресурс]. — URL: http://www.arealidea.ru/articles/tekhnologii-i-algoritmy-dlya-sozdaniya-dopolnennoy-realnosti/ (Дата обращения 17.05.2016). Загл. с экр. Яз. рус.

11 4 Point OpenCV getPerspective Transform Example[Электронный ресурс]. — URL: http://www.pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/ (Дата обращения 19.05.2016). Загл. с экр. Яз. англ.