

МИНОБРНАУКИ РОССИИ

**Федеральное государственное автономное образовательное
учреждение высшего образования
«Южный федеральный университет»**

**Институт математики, механики и компьютерных наук
им. И.И. Воровича**

Шундрикова Александра Владимировна




**УПРАВЛЕНИЕ РОБОТОТЕХНИЧЕСКИМ КОМПЛЕКСОМ
С ПОМОЩЬЮ ЖЕСТОВ**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
по направлению 02.03.02 — Фундаментальная информатика
и информационные технологии**

**Научный руководитель —
ст. преп. Пучкин Максим Валентинович**



Руководителем направлен  *Тема В С*

Ростов-на-Дону – 2018

ОТЗЫВ

о выпускной квалификационной работе А.В. Шундриковой «Управление робототехническим комплексом с помощью жестов»

Целью работы являлась реализация системы анализа видеопотока с целью распознавания и классификации жестов рук, а также выполняющей определённые действия в качестве ответной реакции. Данная система предполагается к использованию совместно с робототехническим комплексом в рамках открытых мероприятий (экскурсий, выставок). Режим работы – демонстрация возможностей по управлению робототехническим комплексом с помощью жестов рук, например, движение робота или синтез речи. В качестве источника видеопотока предполагалось использование камеры смартфона или веб-камеры с достаточным качеством изображения.

В рамках работы были рассмотрены два подхода к выделению рук на изображениях – цветовая сегментация изображения и каскадный классификатор Хаара. Сформировано множество жестов для распознавания, построена обучающая выборка. В качестве методов классификации жестов были рассмотрены цветовая сегментация изображения, и свёрточные нейронные сети.

Результатом работы стало приложение, выполняющее детектирование и классификацию жестов рук, реализованное на языке Python (с использованием соответствующих библиотек) для ОС Linux.

Работа над поставленными задачами велась довольно регулярно, при этом А.В. Шундрикова продемонстрировала высокий уровень компетенций при работе с литературой, навыки разработки и реализации программных комплексов.

Считаю, что работа А.В. Шундриковой «Управление робототехническим комплексом с помощью жестов» соответствует требованиям, предъявляемым к выпускным квалификационным работам студентов бакалавриата, и заслуживает оценки «отлично».

Ст. преп.

каф. прикладной математики и
программирования



М.В. Пучкин

Задание на выпускную квалификационную работу бакалавра

Направление подготовки: фундаментальная информатика и информационные технологии

Образовательная программа бакалавриата:

Студент: А.В. Шундрикова

Научный руководитель: ст. преподаватель М.В. Пучкин

Год защиты: 2018

Тема работы: Управление робототехническим комплексом с помощью жестов.

Цель работы: Реализовать систему, выполняющую анализ видеопотока с целью распознавания и классификации жестов рук, а также выполняющую определённые действия в качестве ответной реакции. Данная система предполагается к использованию совместно с робототехническим комплексом в рамках открытых мероприятий (экскурсий, выставок). Режим работы – демонстрация возможностей по управлению робототехническим комплексом с помощью жестов рук, например, движение робота или синтез речи. В качестве источника видеопотока рассмотреть веб-камеру с достаточным качеством видео (например, HD).

Задачи работы:

- изучить возможности библиотек компьютерного зрения – OpenCV, AForge.Net и аналогичных;
- рассмотреть подходы к выделению движущихся объектов с целью сегментации анализируемой последовательности изображений (видеопотока) и локализации объектов, представляющих интерес с точки зрения классификации;
- сформулировать требования к множеству распознаваемых жестов – количество, ограничения на размер объекта, скорость выполнения и другие;
- выбрать подходящую модель для решения задачи отслеживания объектов и распознавания жестов, построить тестовое множество, разработать архитектуру системы;
- изучение и подбор средств реализации на основе библиотек компьютерного зрения;
- программная реализация демонстрационной версии системы, тестирование.

Заведующий кафедрой

Научный руководитель

Студент

25 января 2018 г.



Г.А. Угольников



М.В. Пучкин

А.В. Шундрикова



СПРАВКА

о результатах проверки текстового документа на наличие заимствований

Проверка выполнена в системе
Антиплагиат.ВУЗ

Автор работы	Шундрикова Александра Владимировна
Факультет, кафедра, номер группы	ИММ и КН кафедра прикладной математики и программирования группа 9
Тип работы	Выпускная квалификационная работа
Название работы	Управление робототехническим комплексом с помощью жестов
Название файла	Управление робототехническим комплексом с помощью жестов.pdf
Процент заимствования	8,40%
Процент цитирования	1,02%
Процент оригинальности	90,58%
Дата проверки	11:11:23 08 июня 2018г.
Модули поиска	Кольцо вузов; Модуль поиска общеупотребительных выражений; Модуль поиска "ЮФУ"; Модуль поиска перефразирований Интернет; Модуль поиска перефразирований eLIBRARY.RU; Модуль поиска Интернет; Коллекция eLIBRARY.RU; Модуль поиска переводных заимствований; Цитирование; Коллекция РГБ; Сводная коллекция ЭБС

Работу проверил Атаманенко Лариса Федоровна

ФИО проверяющего

Дата подписи

8.06.2018

Зам. директора



Подпись проверяющего

Чтобы убедиться
в подлинности справки,
используйте QR-код, который
содержит ссылку на отчет.



Ответ на вопрос, является ли обнаруженное заимствование
корректным, система оставляет на усмотрение проверяющего.
Предоставленная информация не подлежит использованию
в коммерческих целях.

Содержание

Введение	3
1. Постановка задачи	5
2. Предметная область	6
3. Основные подзадачи	8
3.1. Предварительная обработка изображения	8
3.2. Извлечение жеста из изображения	8
3.2.1. Цветовая сегментация изображения	9
3.2.2. Каскадный классификатор Хаара	12
3.3. Классификация жестов	14
3.3.1. Дефекты выпуклости	14
3.3.2. Сверточные нейронные сети	15
3.4. Отслеживание движения руки	18
4. Программная реализация и анализ	20
Заключение	31
Список литературы	32

Введение

В настоящее время компьютеры и информационные технологии являются неотъемлемой частью нашей жизни. Они предоставляют нам доступ к огромному количеству информации, расположенной в сети Интернет. Сейчас существует большое количество онлайн библиотек, которые содержат всю необходимую литературу для обучения, такую как энциклопедии, исследования и научные работы. Существует возможность обратиться к людям за помощью в том или ином вопросе или, наоборот, подсказать кому-то решение онлайн на таких сайтах, как Stack Overflow. Можно поучаствовать в различных проектах других людей по всему миру, предлагая улучшения исходного кода или начать свой собственный проект в сети благодаря таким системам контроля версий, как Github.

Для IT-специалиста большое значение имеет используемое им программное обеспечение. Современные среды разработки помогают программисту решать задачи быстрее и оптимальнее, указывая на ошибки и показывая подсказки. Во время выполнения данной работы было использовано свободно распространяемое программное обеспечение, такое как текстовый редактор Atom, который был разработан специально для написания кода, язык программирования Python 3.6 и различные библиотеки для работы с изображениями, математическими вычислениями и нейронными сетями.

Подготовленная мной выпускная бакалаврская работа посвящена распознаванию жестов. Распознавание жестов — это тема в области наук компьютерного зрения и обработки естественного языка, целью которой является интерпретация человеческих жестов посредством математических алгоритмов. Распознавание жестов можно рассматривать как способ для компьютеров начать понимать язык человеческого тела, тем самым создавая более богатый мост между машинами и людьми.

Система распознавания жестов играет важную роль во взаимодействиях людей с роботами из-за того, что жесты являются естественным и мощным способом коммуникации, и могут использоваться для дистанционного управления.

Особенностями распознавания жестов являются:

- Более высокая точность;
- Высокая стабильность;
- Экономия времени для управления устройством.

При выполнении работы в рамках семинарских занятий Лаборатории искусственного интеллекта и робототехники ИММиКН им. И.И. Воровича были проведены обсуждения со студентами, имеющими схожие темы научных работ. Во время обсуждений были выдвинуты новые идеи и замечания по поводу реализации, дальнейшего объединения разработок и их использования во время проведения различных открытых мероприятий.

1. Постановка задачи

Целью данной работы является реализация системы, выполняющей анализ видеопотока с целью распознавания и классификации жестов рук, а также выполняющей определенные действия в качестве ответной реакции. Данная система предполагается к использованию совместно с робототехническим комплексом в рамках открытых мероприятий (экскурсий, выставок).

В работе решаются следующие задачи:

- Изучение возможностей библиотек компьютерного зрения, включающих в себя средства обработки изображений и распознавания образов.
- Рассмотрение подходов к выделению движущихся объектов с целью сегментации анализируемой последовательности изображений (видеопотока) и локализации объектов, представляющих интерес с точки зрения классификации.
- Формулировка требования к множеству распознаваемых жестов — количество, ограничения на размер объекта, скорость выполнения и другие.
- Выбор подходящей модели для решения задачи отслеживания объектов и распознавания жестов, построение тестового множества, разработка архитектуры системы.
- Изучение и подбор средств реализации на основе библиотек компьютерного зрения.
- Программная реализация демонстрационной версии системы, тестирование.

2. Предметная область

Управление с помощью жестов позволило преодолеть барьеры и ограничения, приблизив человека на один шаг ближе к индивидуальному взаимодействию с компьютером. В последние годы было проведено много исследований с целью разработки новых устройств и технологий, предполагающих управление жестами [1].

В рамках данной работы под жестами понимаются определенные конфигурации руки. Для реализации системы распознавания был разработан набор из четырех жестов (см. Рис 1).

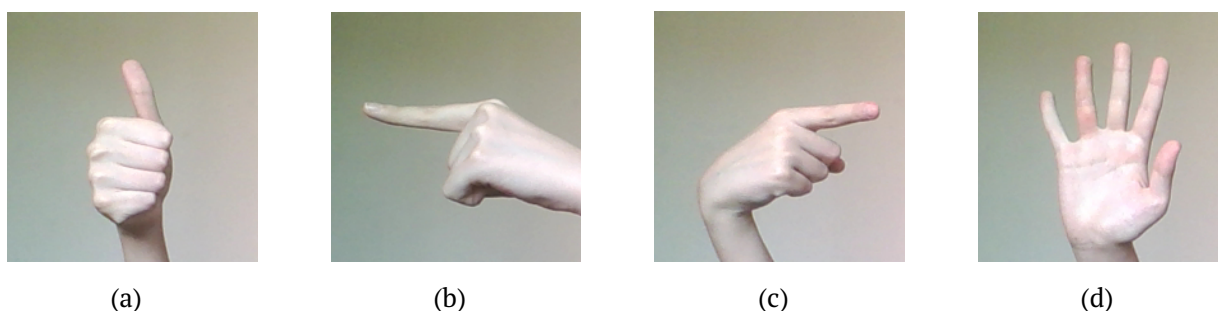


Рисунок 1 — Набор жестов: (a) вперед; (b) влево; (c) вправо; (d) стоп.

Необходимые требования и ограничения:

- расположение руки примерно на расстоянии 1 метра или ближе от камеры;
- отчетливая видимость руки, она не должна перекрываться другими объектами;
- плавное движение руки для более точного отслеживания;
- достаточно хорошее освещение для проведения распознавания;
- видео- или веб-камера с HD разрешением.

Традиционно алгоритм для распознавания жестов в реальном времени состоит из таких этапов [2; 3]:

Регистрация изображения с камеры — захват кадра с камеры для его дальнейшей обработки.

Предварительная обработка изображения — удаление шумов с помощью размытия и морфологических преобразований, а также различные геометрические и цветовые изменения. Необходима для упрощения классификации жеста.

Извлечение жеста — отыскание жеста на изображении и запоминание его месторасположения или извлечение цветов пикселей для создании маски.

Классификация жеста — определение того, к какой команде относится данный жест.

Отслеживание движения руки — поиск объекта в последовательных кадрах. Необходимо для облегчения нахождения нового положения руки в видеопотоке.

3. Основные подзадачи

3.1. Предварительная обработка изображения

Часто входное изображение предварительно обрабатывается для нормализации контрастности, эффектов яркости и удаления шумов. При обработке цветных изображений преобразование цветового пространства (например, RGB в цветовое пространство HSV или YCbCr) может помочь получить лучшие результаты [4].

Размытие изображения полезно для удаления шумов. Оно достигается путем свертывания изображения с ядром фильтра нижних частот. Размытие фактически удаляет из изображения высокочастотный контент (например, шум, ребра). Так края немного размыты после выполнения этой операции (см. Рис. 2) [5].

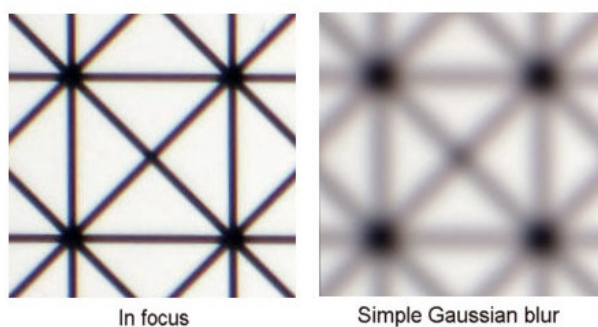


Рисунок 2 — Размытие по Гауссу

3.2. Извлечение жеста из изображения

На входном изображении слишком много дополнительной информации, которая не требуется для классификации. Поэтому первым шагом в классификации изображений является упрощение изображения путем извлечения важной информации, содержащейся в изображении, и исключения ненужной.

3.2.1. Цветовая сегментация изображения

На изображении обычно существует огромная вариация значений цвета пикселей. Выполнив пороговую обработку, можно упростить изображение, получив его маску (см. Рис 3). Далее к маске могут применяться морфологические преобразования [6].



Рисунок 3 — Применение пороговой обработки: (a) исходное изображение; (b) маска.

Морфологические преобразования — это некоторые простые операции, основанные на форме изображения. Обычно они выполняются на двоичных изображениях. Они нуждаются в двух входах, один из которых является исходным изображением, второй — структурирующим элементом или ядром, которое определяет характер работы. Двумя основными морфологическими операциями являются эрозия и дилатация.

Основная идея эрозии — это как эрозия почв, только она разрушает границы объекта переднего плана. Все пиксели около границы будут отброшены в зависимости от размера ядра. Таким образом, толщина или размер объекта переднего плана уменьшается (см. Рис. 4(b)). Это полезно для удаления небольших белых шумов, отсоединения двух связанных объектов и т.д.

Дилатация — это противоположность эрозии, т.е. увеличение размер объекта переднего плана (см. Рис. 4(c)). Обычно в таких слу-

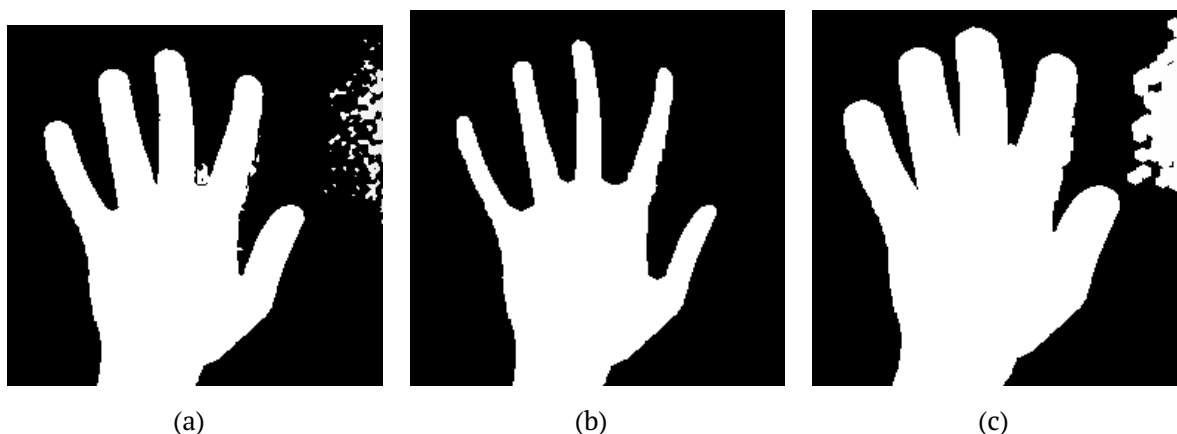


Рисунок 4 — Морфологические преобразования: (а) исходное бинарное изображение; (b) эрозия; (с) дилатация.

чаях, как удаление шума, за эрозией следует дилатация. Потому что эрозия удаляет белые шумы, но также сокращает наш объект. Поэтому мы расширяем его. Поскольку шум ушел, он не вернется, но площадь объекта увеличивается. Это также полезно при соединении сломанных частей объекта.

Все это относится к цветовой сегментации изображения. Однако, для этого предварительно необходимы знания о границах цвета кожи человека. Существуют определенные правила ограничения цвета кожи для трех цветовых пространств — RGB, HSV и YCbCr [4].

Цвет кожи при равномерном дневном освещении определяется как

$$\begin{aligned}
 & (R > 95) \text{ И } (G > 40) \text{ И } (B > 20) \text{ И} \\
 & (\max\{R, G, B\} - \min\{R, G, B\} > 15) \text{ И} \\
 & (|R - G| > 15) \text{ И } (R > G) \text{ И } (R > B)
 \end{aligned} \tag{1}$$

в то время как цвет кожи под фонариком или боковым освещением определяется как

$$\begin{aligned}
 & (R > 220) \text{ И } (G > 210) \text{ И } (B > 170) \text{ И} \\
 & (|R - G| \leq 15) \text{ И } (R > B) \text{ И } (G > B)
 \end{aligned} \tag{2}$$

Чтобы учесть оба условия, когда это необходимо, используется логическое ИЛИ для объединения как правила (1), так и правила (2). Правило ограничения RGB обозначается как правило А.

Правило А: Уравнение (1) \cup Уравнение (2) (3)

Основываясь на наблюдении, что подпространство Cb-Cr является сильным дискриминантом цвета кожи, были сформулированы 5 ограничивающих правил:

$$Cr \leq 1.5862 \times Cb + 20 \quad (4)$$

$$Cr \geq 0.3448 \times Cb + 76.2069 \quad (5)$$

$$Cr \geq -4.5652 \times Cb + 234.5652 \quad (6)$$

$$Cr \leq -1.15 \times Cb + 301.75 \quad (7)$$

$$Cr \leq -2.2857 \times Cb + 432.85 \quad (8)$$

Правила (4) – (8) объединяются с использованием логического И для получения ограничивающего правила CbCr, обозначенного как правило В.

Правило В: Уравнение (4) \cap Уравнение (5) \cap Уравнение (6) (9)
 \cap Уравнение (7) \cap Уравнение (8)

В пространстве HSV значения оттенка показывают наиболее заметное разделение между областями кожи и не кожи. Были вычислены два уровня отсечения, как границы подпространства Н

$$H < 25 \quad (10)$$

$$H > 230 \quad (11)$$

где оба правила объединяются логическим ИЛИ, чтобы получить правило ограничения Н, обозначенное как правило С.

$$\text{Правило С: Уравнение (10)} \cup \text{Уравнение (11)} \quad (12)$$

После этого каждый пиксель, который выполняет правило А, правило В и правило С классифицируется как пиксель цвета кожи:

$$\text{Правило А} \cap \text{Правило В} \cap \text{Правило С} \quad (13)$$

Однако, при использовании метода сегментации могут возникнуть проблемы при непостоянном или плохом освещении. Также на изображении могут встречаться другие объекты с идентичными границами цвета. Данный метод может быть применен в совокупности с другими способами обнаружения и выделения объектов, например, с детекторами признаков.

3.2.2. Каскадный классификатор Хаара

Еще одним способом извлечения жестов является детектор признаков, который может определить жест по форме или другим особенностям. Одним из самых известных является классификатор Хаара.

Каскадный классификатор Хаара [7] — это подход, основанный на механизме обучения, каскадная функция обучается из множества положительных и отрицательных изображений. Затем он используется для обнаружения объектов на других изображениях.

Первоначально алгоритм требует много положительных и отрицательных изображений для обучения классификатора. Затем нужно извлечь из них признаки. Для этого используются признаки Хаара (см. Рис 5). Каждый признак представляет собой одно значение, полученное путем вычитания суммы пикселей под белым прямоугольником из суммы пикселей под черным прямоугольником.

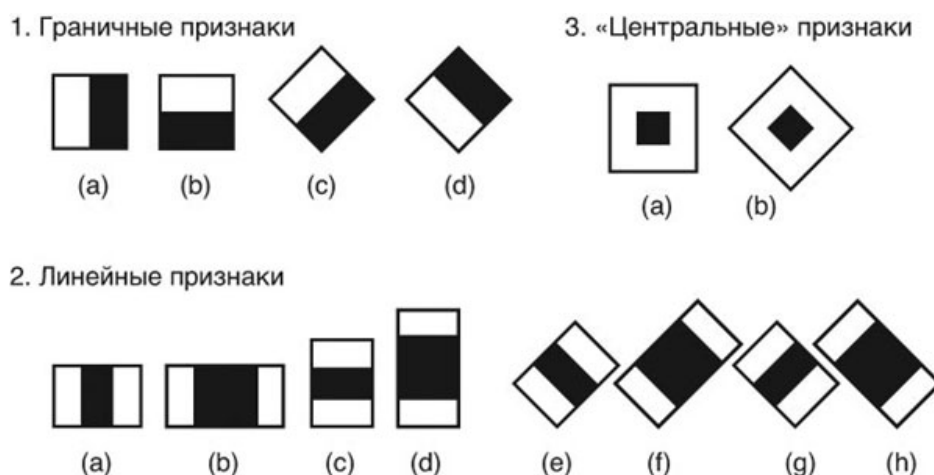


Рисунок 5 — Признаки Хаара

Каждый признак применяется на всей обучающей выборке. Для каждого признака находится лучший порог, который будет относить изображения к положительным или отрицательным. Очевидно, что будут ошибки, поэтому выбираются признаки с минимальной частотой ошибок, что означает, что они являются признаками, которые наиболее точно классифицируют изображения с объектом и без.

Конечный классификатор представляет собой взвешенную сумму этих слабых классификаторов. Он называется слабым, потому что он сам по себе не может классифицировать изображение, но вместе с другими образует сильный классификатор.

На изображении большая часть — это область без объекта. Если часть изображения является такой областью, она отбрасывается и не обрабатывается снова. Таким образом, больше времени отводится на проверку возможных областей объекта.

Для этого была представлена концепция каскада классификаторов. Вместо применения всевозможных признаков на части изображения, признаки группируются на разные этапы и применяются один за другим. (Обычно первые несколько этапов будут содержать очень много признаков). Если часть изображения отбрасывается уже на первом этапе, то на нем остальные признаки не рассматриваются. Если она пройдет проверку, то применяется второй этап признаков и про-

цесс продолжается. Часть изображения, которая проходит все этапы, представляет собой область объекта.

3.3. Классификация жестов

После того, как изображение было упрощено и важная информация была выделена (например, получена маска руки), можно переходить к следующему этапу – классификации полученного жеста.

3.3.1. Дефекты выпуклости

Одним из вариантов классификации жестов может быть нахождение и подсчет дефектов выпуклости (convexity defects) [8].

Первым этапом в нахождении дефектов является выделение контура руки. Контур можно определить как кривую, соединяющую все непрерывные точки (вдоль границы), имеющие одинаковый цвет или интенсивность. Контурные являются полезным инструментом для анализа формы, обнаружения и распознавания объектов. Для лучшей точности используются двоичные изображения. Поэтому, прежде чем находить контур, применяют пороговую обработку, т.е. получают маску. Контуров на полученной маске может быть несколько. Например, если на изображении были другие объекты со схожим цветом. Обычно можно задать минимальный и максимальный размер отыскиваемого контура. Или, если известно, что контур руки является самым большим на изображении, то просто выделить максимальный контур из списка всех найденных.

На следующем этапе находят выпуклую оболочку найденного контура. Выпуклая оболочка контура жеста руки представляет собой выпуклый многоугольник, окруженный всеми выпуклыми вершинами в контуре жеста.

Дефекты выпуклости определяют как разность между выпуклой оболочкой и контуром жеста, они содержатся в выпуклой оболочке,

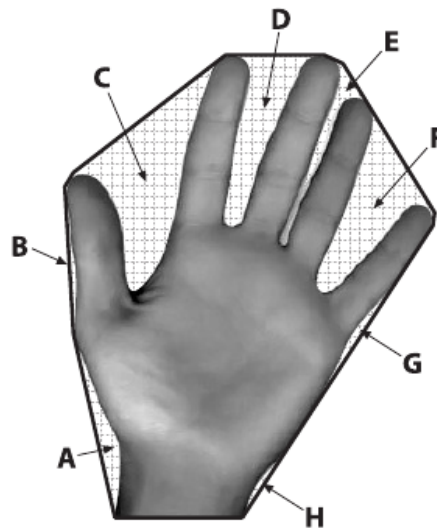


Рисунок 6 — Дефекты выпуклости

но не в области руки. На рисунке 6 области А – Н являются дефектами выпуклости. Структура данных каждого из дефектов выпуклости содержит три компонента: начальную точку контура, конечную точку контура и вогнутую точку контура.

На рисунке 6 можно увидеть, что кончики пальца тесно связаны с дефектами выпуклости, они близки к начальным и конечным точкам. Таким образом, можно обнаружить кончики пальцев, используя дефекты выпуклости.

Подсчитав количество дефектов выпуклости, а именно количество одного из его компонентов (например, вогнутые точки) можно классифицировать жест. Минусом в этом подходе является то, что количество дефектов может быть одинаковым для нескольких жестов. Его можно применить, если набор жестов ограничивается жестами, связанными с количеством пальцев или в совокупности с другим методом.

3.3.2. Сверточные нейронные сети

Другим способом классификации жестов может быть подход, использующий сверточные нейронные сети (CNN). CNN представляют

собой особый вид многослойных нейронных сетей. Как и почти все другие нейронные сети, они обучаются алгоритмом обратного распространения ошибки. Отличаются они от обычных нейронных сетей в архитектуре (см. Рис 7) [9].

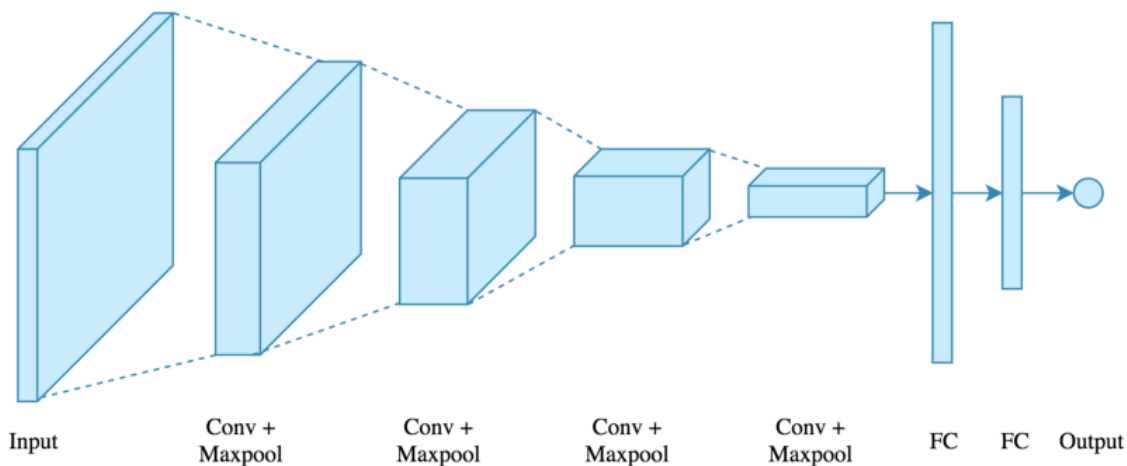


Рисунок 7 — Архитектура свёрточной нейронной сети.

CNN сравнивает изображения по частям. Части, которые она отыскивает, называются признаками. Каждый признак похож на мини-изображение — маленький двумерный массив значений. Признаки соответствуют общим аспектам изображений.

Когда на вход подается новое изображение, CNN не знает точно, где эти признаки будут совпадать, поэтому она проверяет в любой возможной позиции. При вычислении соответствия признака по всему изображению выполняют операцию свертки [10].

Чтобы вычислить совпадение признака с патчем изображения, просто умножают каждый пиксель в признаке на значение соответствующего пикселя в изображении. Затем полученные значения складываются и делятся на общее число пикселей в этом признаке. Если оба пикселя белые (значение 1), то $1 * 1 = 1$. Если оба являются черными, то $(-1) * (-1) = 1$. В любом случае каждый совпадающий пиксель приводит к 1. Аналогично, любой несовпадающий приводит к -1. Если все пиксели в признаке совпадают, то их суммирование и деление на общее количество пикселей дает 1. Аналогичным образом, если ни

один из пикселей в признаке не соответствует патчу изображения, тогда ответ равен -1.

Чтобы завершить свертку, процесс повторяется, подбирая признак к каждому из всех возможных патчей изображения. Можно получить значение каждой свертки и сделать из них новый двумерный массив, основываясь на том, где на изображении находится каждый патч. Эта карта совпадений также является отфильтрованной версией оригинального изображения.

Следующий шаг — полностью повторить процесс свертки для каждого из других признаков. Результатом является набор отфильтрованных изображений. Удобно рассматривать весь этот набор операций свертки как один шаг обработки. В CNN это называется слоем свертки.

Еще один сильный инструмент, используемый CNN, называется пулингом. Пулинг — это способ взять большие изображения и сжать их, сохранив в них самую важную информацию. Он предполагает под собой создание небольшого окна на изображении и получение максимального значения из этого окна на каждом шаге.

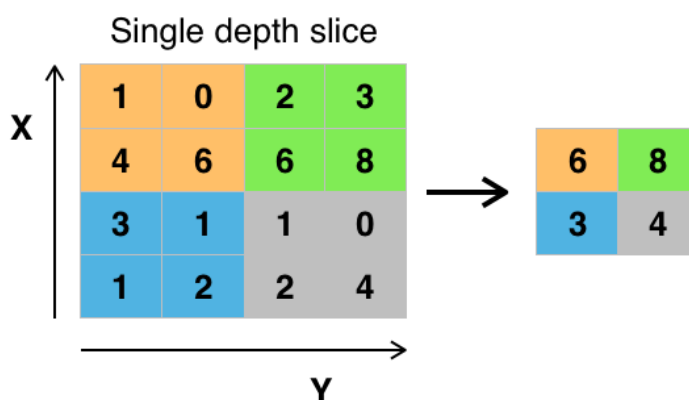


Рисунок 8 — Пулинг с функцией максимума и фильтром 2×2 .

Пулинг сохраняет главное значение совпадения каждого признака в окне. Это означает, что неважно, где именно признак совпадает, если он совпадает где-то внутри окна. Результатом этого является

то, что CNN могут найти, есть ли признак в изображении, не беспокоясь о том, где он находится. Слой пулинга — это всего лишь операция выполнения пулинга на изображениях. На выходе будет то же количество изображений, но у каждого будет меньше пикселей.

Небольшую, но важную роль в CNN играет функция активации ReLU (Rectified Linear Unit). Везде, где присутствует отрицательное число, она заменяет его на 0. Это помогает удерживать полученные значения от застревания около 0 или увеличения до бесконечности.

Существует еще один тип слоев — полносвязные. Полносвязные слои принимают отфильтрованные изображения и переводят их в голоса за определенный класс. Полносвязные слои являются основным строительным блоком традиционных нейронных сетей. Вместо рассмотрения входов как двумерных массивов, они рассматриваются как один список, и все обрабатываются одинаково. Каждое значение получает свой собственный голос о том, относится ли текущий образ к тому или иному классу.

3.4. Отслеживание движения руки

С каждым новым кадром, положение руки может меняться, поэтому для облегчения нахождения нового положения руки используют отслеживание. Проще говоря, отслеживание — это поиск объекта в последовательных кадрах видеопотока.

Целью отслеживания является нахождения объекта в текущем кадре, подразумевая, что объект успешно отслеживается во всех (или почти всех) предыдущих кадрах.

Поскольку объект отслеживается до текущего кадра, то известно как он двигался ранее. Другими словами, известны параметры модели движения. Модель движения — это местоположение и скорость (скорость + направление движения) объекта в предыдущих кадрах.

Также известно, как объект выглядит в каждом из предыдущих кадров, поэтому можно построить модель внешнего вида, которая за-

поминает то, как выглядит объект. Эта модель внешнего вида может использоваться для поиска в малой окрестности местоположения, предсказанного моделью движения, для более точного прогнозирования местоположения объекта.

Если объект был очень прост и не сильно менял внешний вид, можно использовать простой шаблон в качестве модели внешнего вида и искать этот шаблон. Но появление объекта может резко измениться. Чтобы решить эту проблему, во многих современных трекерах эта модель внешнего вида является классификатором, который обучается во время выполнения программы.

4. Программная реализация и анализ

В рамках данной выпускной квалификационной работы была разработана система распознавания жестов. Программная реализация была написана на языке Python с использованием таких сторонних библиотек, как OpenCV, NumPy, scikit-learn и Keras.

Первичная обработка изображения

В программу каждый раз поступает кадр из видеопотока, который в первую очередь необходимо обработать. Для обработки изображений была использована библиотека OpenCV.

Удаление шума было сделано с помощью функции `cv2.GaussianBlur()`. Фильтрация Гаусса очень эффективна при удалении гауссовского шума из изображения.

Также изображение в цветовом пространстве RGB было преобразовано в HSV с помощью функции `cv2.cvtColor()`, потому что HSV имеет большее отношение к восприятию цвета человеком, чем RGB.

Извлечение жеста из изображения

Для извлечения жеста из изображения была использована совокупность таких методов, как каскадный классификатор Хаара, метод k-средних и пороговая обработка.

Каскадный классификатор `cv2.CascadeClassifier()` был обучен для нахождения сжатого кулака на входном кадре из видеопотока. После его обнаружения, изображение обрезаются до региона, в котором был обнаружен кулак и подается на вход функции `KMeans()` из библиотеки `scikit-learn`.

Метод k-средних кластеризует данные, пытаясь разделить выборку на k групп, сводя к минимуму критерий, известный как инерция или внутрикластерная сумма квадратов. Он хорошо масштаби-

руется для большого количества образцов и используется во многих областях. В данном случае он используется для разделения пикселей обрезанного изображения на кластеры по цвету, чтобы найти наиболее часто встречающиеся цвета.

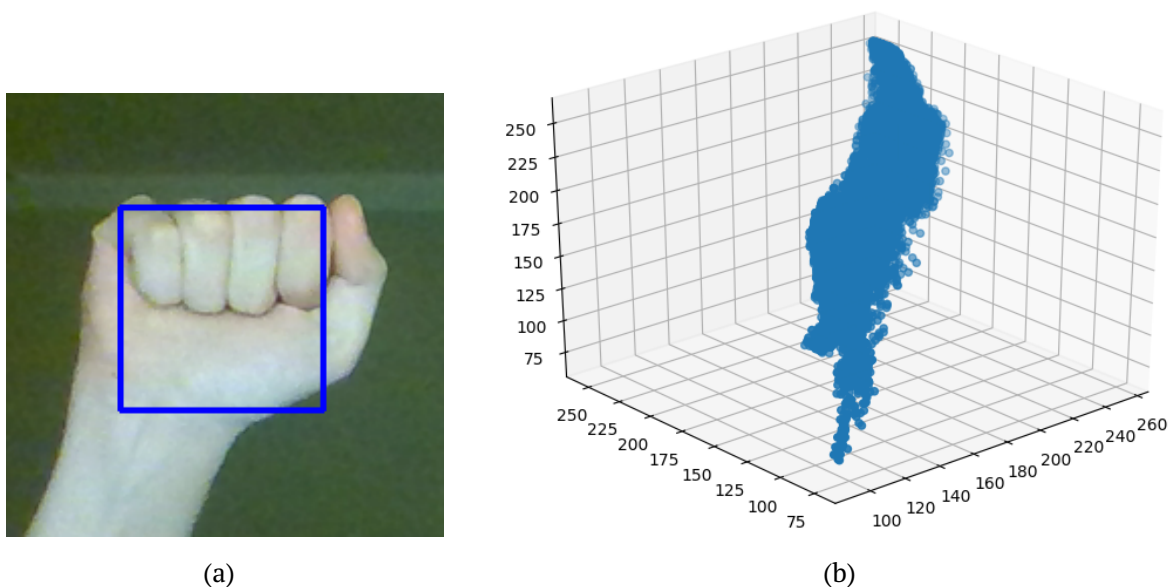


Рисунок 9 — (a) обнаружение руки классификатором Хаара; (b) кластеризация методом k-средних.

После нахождения этих значений цветов, можно подобрать подходящие границы для пороговой обработки исходного изображения и с помощью функции `cv2.inRange()` получить его маску.



Рисунок 10 — Открытие: (a) исходная маска; (b) маска после морфологических преобразований.

Далее маска проходит еще один этап обработки. Обработка заключается в применении морфологических преобразований к изображению. Для удаления шума используется операция открытия, т.е. последовательно применяются операции эрозии (`cv2.erode()`) и дилатации (`cv2.dilate()`).

В процессе обнаружения кулака или выбора наиболее частого цвета могут возникать ошибки, поэтому в программу были добавлены специальные слайдеры, с помощью которых можно регулировать значения порогов вручную (см. Рис 11).

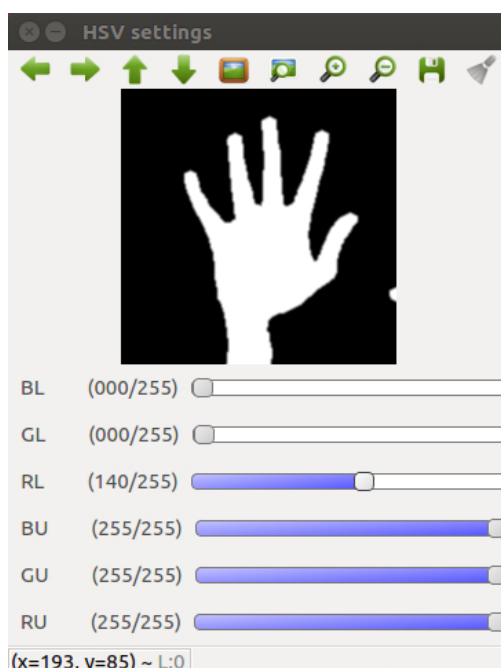


Рисунок 11 — Слайдеры для управления границами значений порогов

Классификация жеста

Дефекты выпуклости

В первой версии программы для выполнения задачи классификации был реализован подход, использующий дефекты выпуклости.

После получения маски кадра, на ней отыскивались все возможные контуры с помощью функции `cv2.findContours()`. После чего среди полученного списка контуров отыскивался максимальный, который и должен был соответствовать контуру руки.

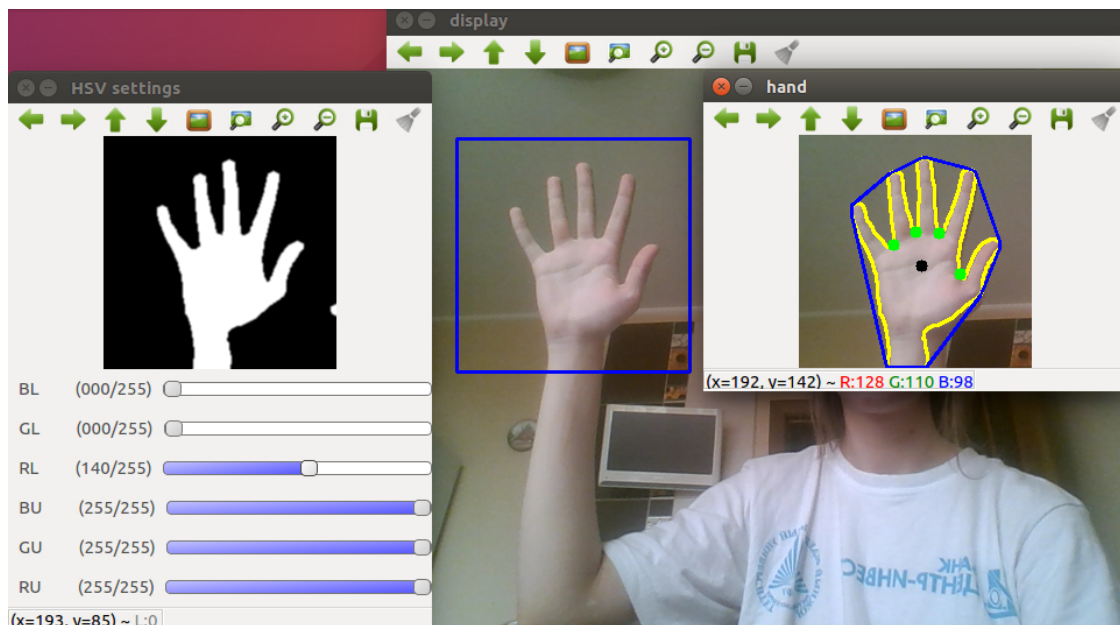


Рисунок 12 — Работа первой версии программы с использованием эффектов выпуклости.

С помощью функции `cv2.convexHull()` была получена выпуклая оболочка найденного ранее контура, а функция `cv2.convexityDefects()` возвращала список всех дефектов выпуклости полученной оболочки. Программа подсчитывала вогнутые точки и с помощью них определяла количество пальцев (см. Рис 12).

Данная версия программы не стала в дальнейшем финальной из-за ограниченного спектра возможных распознаваемых жестов. Набор жестов представляющий собой просто количество пальцев показался слишком тривиальным и недостаточно естественным для обозначения команд робота. Но как вариант, можно совместить ее с другим методом классификации, чтобы получить большую точность определения того или иного класса жеста.

Сверточные нейронные сети

Во второй версии программы было принято решение использовать сверточные нейронные сети. CNN предназначены для распознавания визуальных шаблонов непосредственно из пиксельных изображений с минимальной предварительной обработкой. В программе была реализована CNN при помощи библиотеки Keras [11; 12]. Она обеспечивает очень простой рабочий процесс для обучения и оценки сети.

Для обучения сети был разработан набор жестов (см. Рис 13), подразумевающий такие команды как:

1. движение вперед;
2. поворот влево;
3. поворот вправо;
4. прекращение движения.

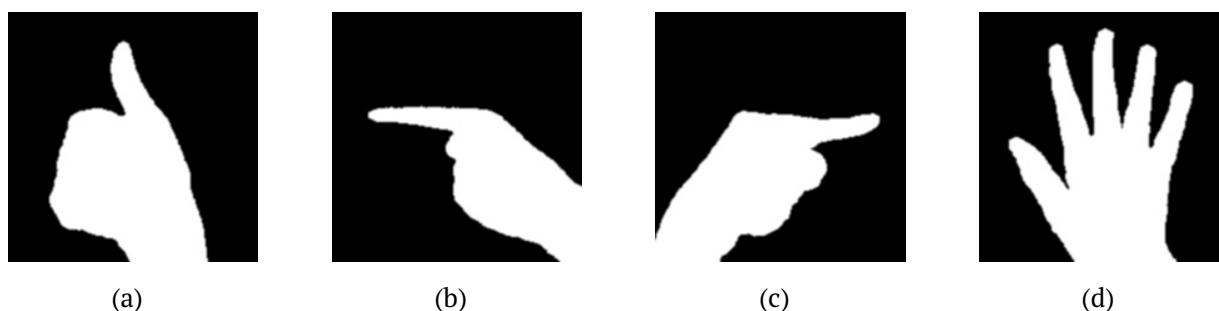


Рисунок 13 — Команды: (a) вперед; (b) влево; (c) вправо; (d) стоп.

Каждая команда/жест соответствует определенному классу, который CNN должна распознать. Для каждого класса были подготовлены по 300 образцов для обучения и 40 образцов для проверки, т.е. всего 1200 изображений в обучающей выборке и 160 в тестовой. Каждое изображение является маской. Это упрощает задачу моделирования сети. Размер изображений был выбран 54×54 .

CNN была реализована в виде простой последовательной (sequential) модели с тремя сверточными слоями и тремя слоями пулинга (см. Табл 1). После прохождения тензора через сверточные слои

Таблица 1 — Модель сверточной нейронной сети.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 52, 52, 32)	320
activation_1 (Activation)	(None, 52, 52, 32)	0
max_pooling2d_1 (MaxPooling2)	(None, 26, 26, 32)	0
conv2d_2 (Conv2D)	(None, 24, 24, 32)	9248
activation_2 (Activation)	(None, 24, 24, 32)	0
max_pooling2d_2 (MaxPooling2)	(None, 12, 12, 32)	0
conv2d_3 (Conv2D)	(None, 10, 10, 64)	18496
activation_3 (Activation)	(None, 10, 10, 64)	0
max_pooling2d_3 (MaxPooling2)	(None, 5, 5, 64)	0
flatten_1 (Flatten)	(None, 1600)	0
dense_1 (Dense)	(None, 64)	102464
activation_4 (Activation)	(None, 64)	0
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 4)	260
activation_5 (Activation)	(None, 4)	0

он выравнивается в вектор и проходит через плотные (полносвязные) слои.

Первый слой модели conv2d_1 — это сверточный слой, который состоит из 32 обучаемых фильтров с шириной и высотой 5 пикселей. Нам не нужно определять содержимое этих фильтров, поскольку модель будет изучать строительные фильтры, как бы «наблюдая» некоторые типы визуальных признаков входных изображений, таких как край или кривая изображения. 32 фильтра первого слоя должны выглядеть так на рисунке 14.

В первом сверточном слое каждый из 32 фильтров подключается к входным изображениям и создает двумерную карту признаков для каждого изображения. Таким образом, из первых выходов сверточного слоя имеется 32×1200 (количество входных изображений) = 38400 карт признаков. Можно визуализировать выход, используя случайное изображение из 1200 входов (см. Рис 15).

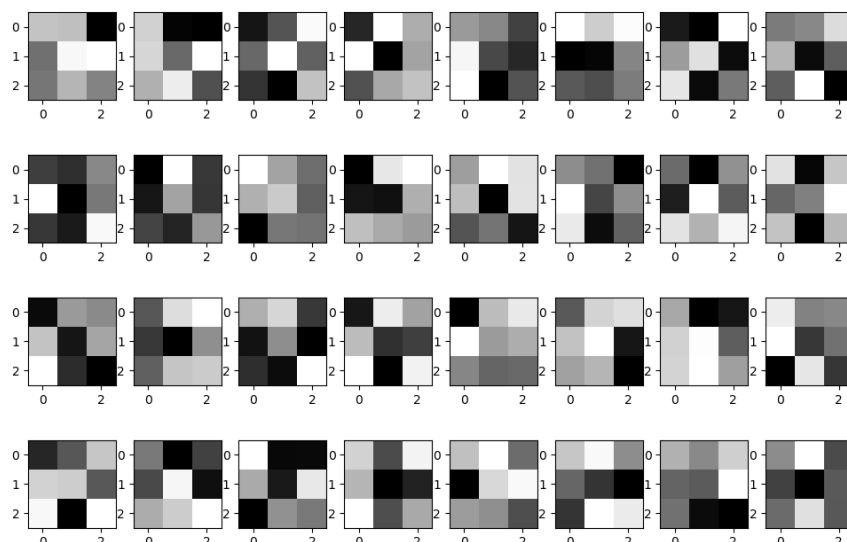


Рисунок 14 — Фильтры первого сверточного слоя.

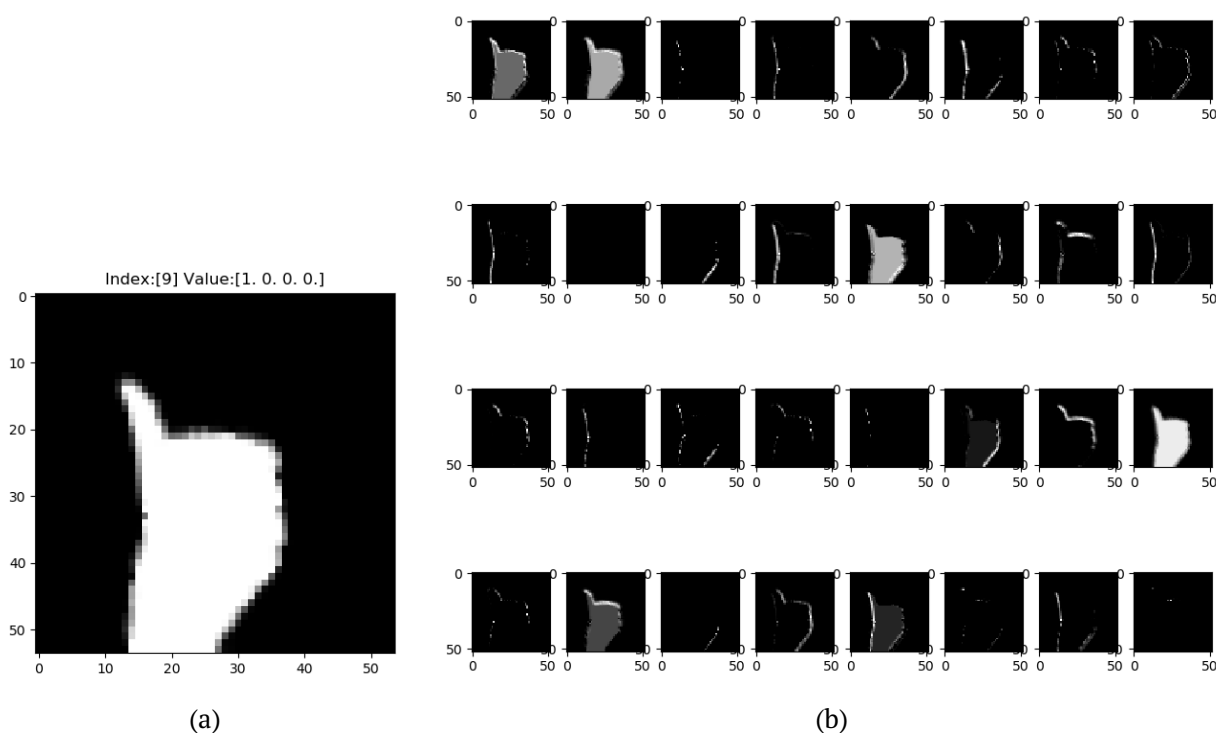


Рисунок 15 — (a) случайное изображение из 1 200 входов; (b) 32 карты признаков из первого слоя.

Сразу после выполнения нашего первого слоя уже есть 38 400 выходов в виде карт признаков. Общепринятой практикой является применение слоя пулинга. Таким образом, мы можем подвыбрать на-

ши результаты, чтобы уменьшить параметры и вычисления в нашей модели. В нашем случае используется 2×2 размер пула. Размер изображения резко уменьшается (см. Рис 16). Поскольку сохраняются важные данные, такие как наличие признака, а не его точное местоположение, можно избежать переобучения.

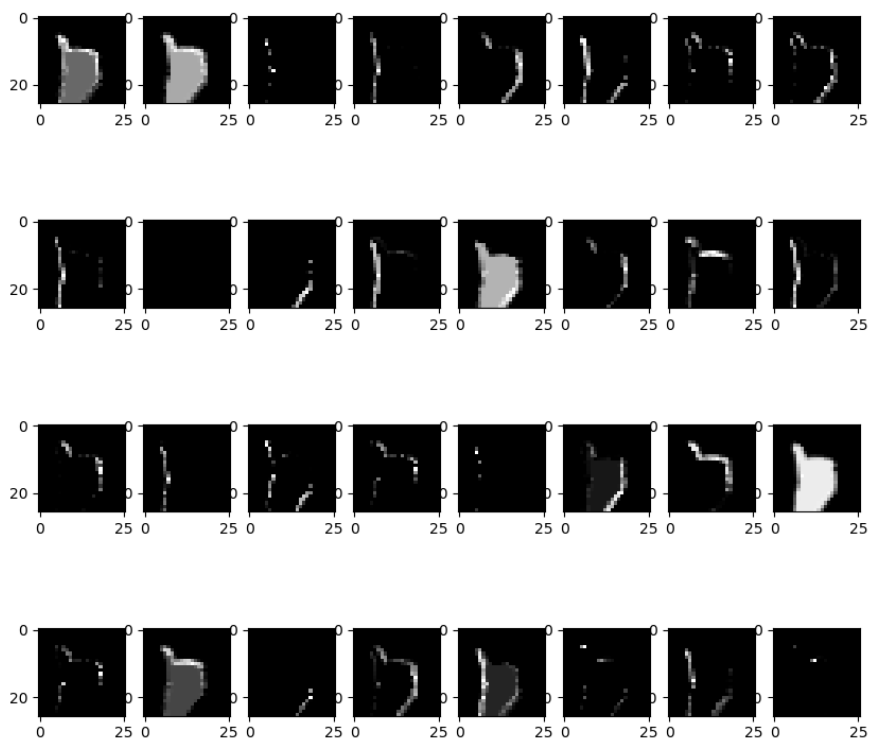


Рисунок 16 — Карты признаков после пулинга.

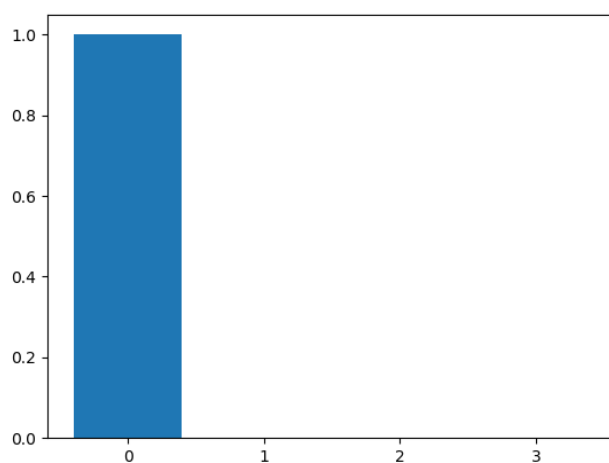


Рисунок 17 — Результат классификации.

После прохождения через сверточные слои и слои пулинга карты проходят несколько полносвязных слоев. Затем модель классифицирует входные данные на последнем слое (см. Рис 17).

Обучение было выполнено за 20 эпох. В результате точность распознавания на обучающей выборке достигла 98%, потеря – 0.55, на тестовой выборке точность – 99%, потеря – 0.011 (см. Рис 18).

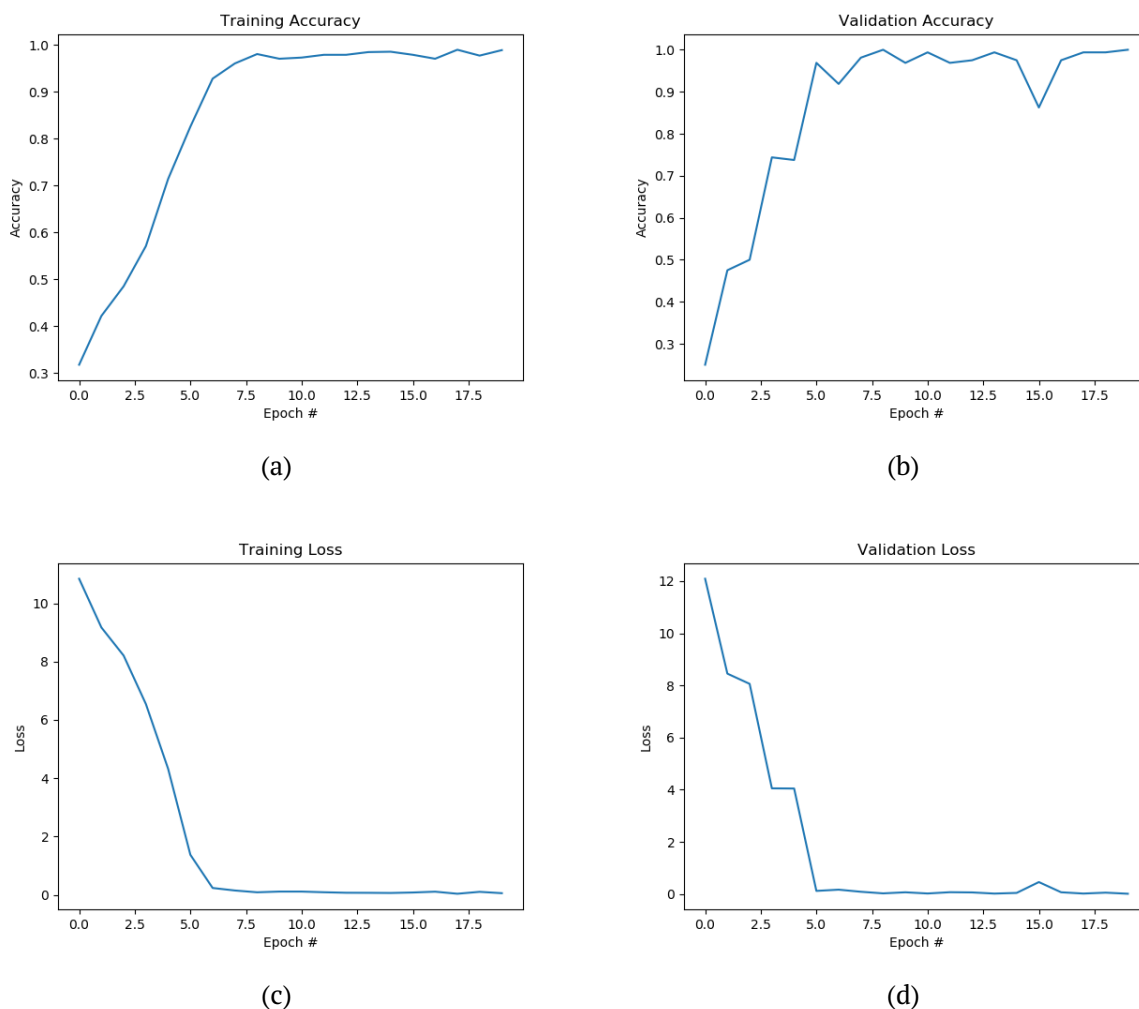


Рисунок 18 — Графики изменения качества модели: (а) точность на этапе обучения; (б) точность на этапе проверки; (с) потери на этапе обучения; (д) потери на этапе проверки.

Отслеживание движения руки

Программа подразумевает возможность отслеживания движения руки. Оно было реализовано с помощью Tracking API библиотеки OpenCV. В данном API реализованы несколько алгоритмов отслеживания объектов: BOOSTING, MIL, KCF, TLD, MEDIANFLOW и GOTURN.

На практике для отслеживания движения руки лучше всего подошел KCF трекер. В использовании данного трекера практически не возникало ошибок, он оказался наиболее точным и быстро реагирующим на передвижение. Также неплохие результаты показал трекер MEDIANFLOW при достаточно спокойном передвижении руки. Другие трекеры оказались недостаточно точными для выполнения требуемой задачи.

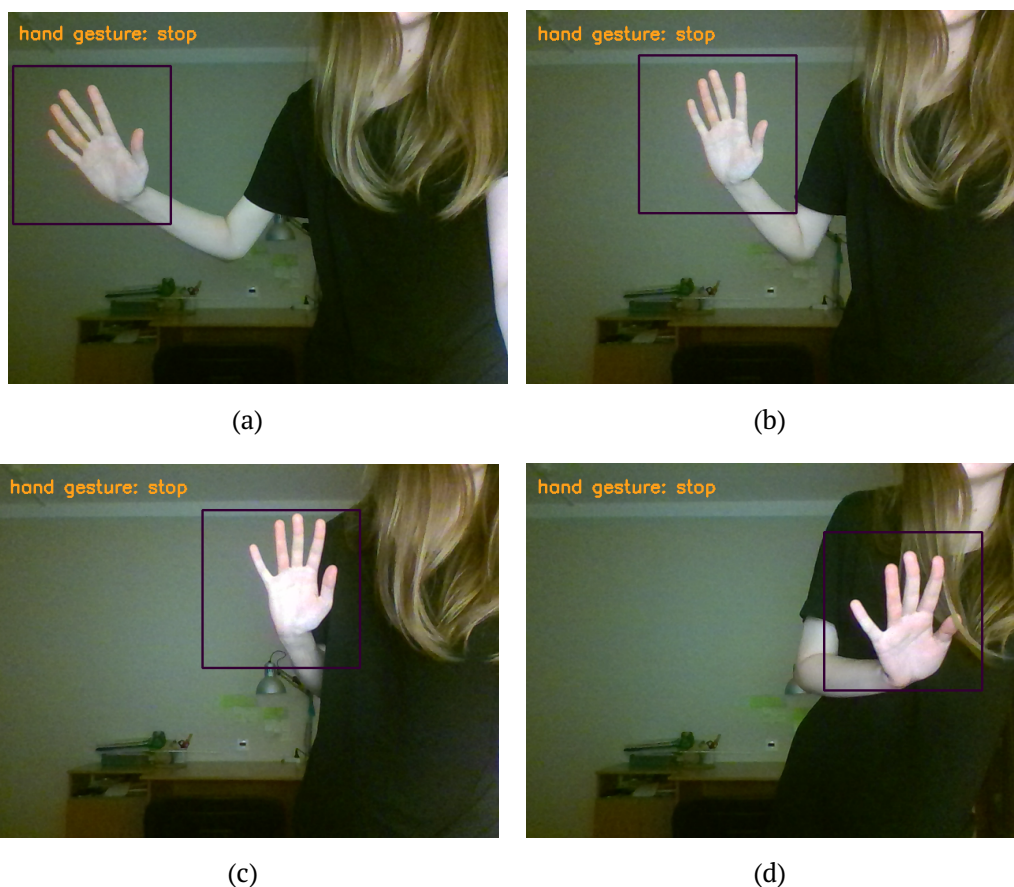


Рисунок 19 — Процесс отслеживания движения руки.

В результате для реализации был выбран трекер KCF. Во время работы программы необходимо поместить руку в выделенное ограничивающее поле и нажать определенную клавишу клавиатуры, чтобы начать отслеживание (см. Рис 19).

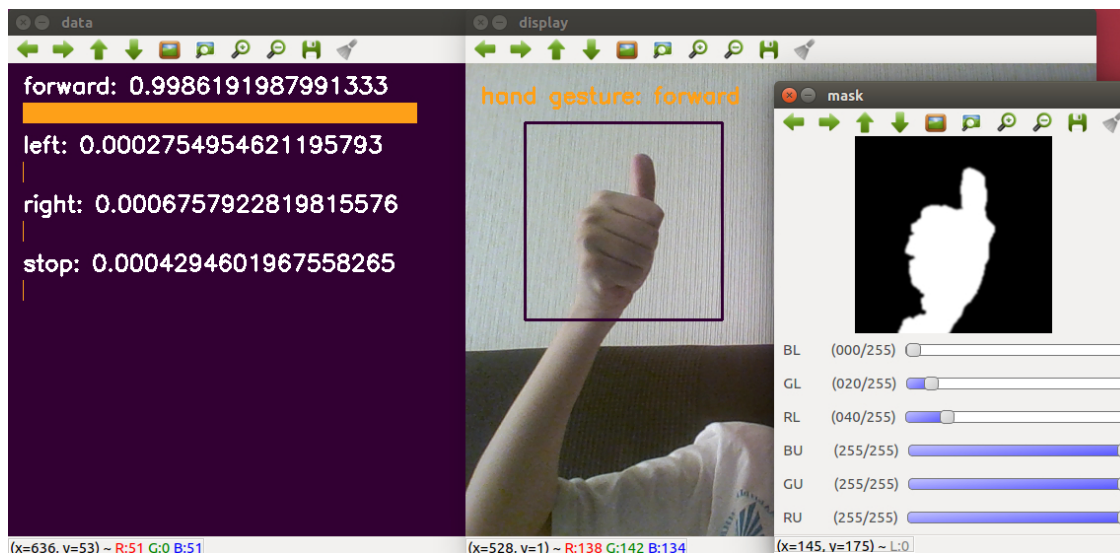


Рисунок 20 — Работа второй версии программы с распознаванием и отслеживанием движения.

Заключение

В процессе выполнения работы были решены следующие задачи:

- Изучены основные методы для решения задачи распознавания жестов рук
- Выбрана модель для решения задачи отслеживания и распознавания и разработана архитектура системы
- Реализована система, выполняющая анализ видеопотока с целью распознавания жестов рук
- Проанализирована и протестирована работа разработанной системы

Список литературы

1. *Mahmoudi F., Parviz M.* Visual Hand Tracking Algorithms. — 1993. — URL: https://www.researchgate.net/publication/224636576_Visual_Hand_Tracking_Algorithms.
2. *Manresa-Yee C., Varona J., Perales F. J.* Hand tracking and gesture recognition for human-computer interaction. — 2005. — URL: https://www.researchgate.net/publication/39087103_Hand_Tracking_and_Gesture_Recognition_for_Human-Computer_Interaction.
3. Gesture Recognition for Robotic Control Using Deep Learning / C. Kawatsu [и др.]. — 2017. — URL: https://www.researchgate.net/publication/320084139_Gesture_Recognition_for_Robotic_Control_Using_Deep_Learning.
4. *Abdul Rahman N. A. bin, Wei K. C., See J.* RGB-H-CbCr Skin Colour Model for Human Face Detection. — URL: <http://academic.aau.am/Skhachat/Public/Papers%20on%20Face%20Detection/RGB-H-CbCr%20Skin%20Colour%20Model%20for%20Human%20Face%20Detection.pdf>.
5. *Shaikh N. R.* Smoothing of a noisy image using different low pass filters. — 2017. — URL: <http://www.ijcstjournal.org/volume-5/issue-2/IJCST-V5I2P50.pdf>.
6. OpenCV. Morphological Transformations. — URL: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html#morphological-ops.
7. *Viola P., Jones M.* Rapid object detection using a boosted cascade of simple features. — 2001. — URL: <https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf>.

8. *Xu Y., Park D.-W., Pok G.* Hand Gesture Recognition Based on Convex Defect Detection. — 2017. — URL: <https://pdfs.semanticscholar.org/ee87/b0b46ed7a31ef90ee672c08a22e028e4537c.pdf>.
9. *Nielsen M.* Neural Networks and Deep Learning. — 2017. — URL: <http://neuralnetworksanddeeplearning.com/chap6.html>.
10. CS231n: Convolutional Neural Networks for Visual Recognition. — URL: <http://cs231n.github.io/convolutional-networks/>.
11. Keras Documentation. — URL: <https://keras.io/getting-started/sequential-model-guide/>.
12. *Франсуа Ш.* Глубокое обучение на Python. — 2018. — URL: <https://books.google.ru/books?id=97ZaDwAAQBAJ&printsec=frontcover&hl=ru#v=onepage&q&f=false>.