|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | ИУ «Информатика, искусственный интеллект и системы управления» |

|  |  |
| --- | --- |
| КАФЕДРА | ИУ-1 «Системы автоматического управления» |

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

|  |
| --- |
|  |
|  |
| ***Методы и технологии управления курсором*** |
| ***мыши с помощью распознавания жестов*** |
|  |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ1-81Б |  | 25/04/2022 |  | Д.И. Юдаков |
|  | (Группа) |  | (Подпись, дата) |  | (И.О. Фамилия) |

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель | 25/04/2022 |  | К.В. Парфентьев |
|  | (Подпись, дата) |  | (И.О. Фамилия) |

*2022 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Заведующий кафедрой | | | | | | | ИУ-1 | | |
|  | | | | | | | (Индекс) | | |
|  |  | | | |  | К.А. Неусыпин | | | |
|  | | | | | | (И.О. Фамилия) | | | |
|  | « | 08 | » | февраля | | | 20 | 22 | г. |

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| по теме | | Методы и технологии управления курсором мыши с помощью | | |
| распознавания жестов | | | | |
|  | | | | |
| Студент группы | ИУ1-81Б | |  | |
| Юдаков Дмитрий Игоревич | | | | |
| (Фамилия, имя, отчество) | | | | |
| Направленность НИР (учебная, исследовательская, практическая, производственная, др.) | | | | |
| учебная | | | | |
| Источник тематики (кафедра, предприятие, НИР) | | | | кафедра |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| График выполнения НИР: | | 25% к |  | н., 50% к |  | н., 75% к |  | н., 100% к |  | н. |
|  | | | | | | | | | | |
| ***Техническое задание*** | Исследовать основные методы и технологии управления курсором | | | | | | | | | |
| мыши с помощью распознавания жестов | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |

***Оформление научно-исследовательской работы:***

|  |  |  |
| --- | --- | --- |
| Расчетно-пояснительная записка на | ***27*** | листах формата А4. |
| Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.) | | |
| оформление графического материала в НИР не предусмотрено | | |
|  | | |
|  | | |
|  | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Дата выдачи задания | « | 08 | » | февраля | 20 | 22 | г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Руководитель НИР** |  | 08/02/2022 |  | К.В. Парфентьев |
|  |  | (Подпись, дата) |  | (И.О. Фамилия) |
|  |  |  |  |  |
| **Студент** |  | 08/02/2022 |  | Д.И. Юдаков |
|  |  | (Подпись, дата) |  | (И.О. Фамилия) |

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 1](#_Toc101299498)

[ВВЕДЕНИЕ 2](#_Toc101299499)

[1. МЕТОДЫ РАСПОЗНАВАНИЯ ЖЕСТОВ 3](#_Toc101299500)

[1.1. Распознавание при помощи вычисления расстояния между точками 3](#_Toc101299501)

[1.2. Распознавание жестов при помощи сигналов тензодатчиков 4](#_Toc101299502)

[1.3. Автоматическое преобразование жестов русской ручной азбуки в текстовый вид [1]. 4](#_Toc101299503)

[1.4. Распознавание жестов при помощи нейронной сети 5](#_Toc101299504)

[2. SINGLE SHOT ДЕТЕКТОР 6](#_Toc101299505)

[2.1. Модель 6](#_Toc101299506)

[2.2. Обучение 8](#_Toc101299507)

[2.3. Аналоги 11](#_Toc101299508)

[2.4. Выводы 13](#_Toc101299509)

[2.5. Библиотека MediaPipe Hands 14](#_Toc101299510)

[3. МЕТОДИКА УПРАВЛЕНИЯ КУРСОРОМ МЫШИ 17](#_Toc101299511)

[3.1. Сбор данных 17](#_Toc101299512)

[3.2. Обучение модели 19](#_Toc101299513)

[3.3. Практическое применение 22](#_Toc101299514)

[ЗАКЛЮЧЕНИЕ 25](#_Toc101299515)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 26](#_Toc101299516)

# ВВЕДЕНИЕ

Вычислительная способность компьютера стала необходимой в жизни современного человека – без неё довольно трудно жить и работать в нынешних реалиях. Умение быстро помочь в решении самых сложных задач начиная от проведения необходимых расчётов до оптимизации управления предприятием высоко оценивается человечеством. Камнем преткновения остаётся интерфейс взаимодействия системы «человек-машина» – оно должно быть максимально упрощено, чтобы люди и машины коммуницировали на уровне естественных средств общения.

Развитие компьютерных технологий получило огромный прирост за последнюю половину века и стремительно увеличивается и по сей день. Они уже способны обрабатывать и анализировать информацию подобно человеку: распознавать текст, изображения, анализировать звуки и мелодии, произносить осмысленные предложения, распознавать голосовые команды и реагировать на прикосновения пальцев. Но одной из самых приоритетных и трудных задач в области информационных технологий и интеллектуальных систем является задача распознавания жестов.

Актуальность рассматриваемой тематики обусловлена возможностью применения предлагаемого подхода для управления объектов без тактильного контакта и голосовой идентификации команд, а также своей простотой с точки зрения конечного пользователя.

В данной работе идёт речь о детектировании кисти руки и распознавание жеста для управления компьютером без использования мыши.

В ходе работы исследуются различные методы и техники детектирования кисти человека и распознавания жеста. Важным требованием для реализации является минимизация времени задержки на обработку каждого кадра для комфортного использования и точного позиционирования курсора на экране.

Поставленной **целью** является создание программы для управления компьютером без мыши с помощью языка высокого уровня Python.

Для достижения цели необходимо решить ряд **задач**:

1. Сделать обзор существующих решений детектирования кисти рук и распознавания жестов.
2. Изучить теоретические аспекты работы Single Shot детектора.
3. Разработать и протестировать методику распознавания жестов для управления курсором мыши.

# 1. МЕТОДЫ РАСПОЗНАВАНИЯ ЖЕСТОВ

Несмотря на актуальность задачи, уже существуют различные способы распознавания жестов. Эти способы можно разделить на несколько групп, объединённых по способу распознавания рук человека. Рассмотрим некоторые из реализаций данных групп.

1.1. Распознавание при помощи вычисления расстояния между точками

Примером реализации данного метода является «Система распознавания жестов в реальном времени с цветными перчатками» (Real-Time Hand-Tracking with a Color Glove), которую создали Роберт Ванг (Robert Y. Wang) и Йован Попович (Jovan Popovic) из лаборатории компьютерных наук и искусственного интеллекта Массачусетского технологического института [14].

Распознавание жестов возможно при надевании цветных перчаток, по которым программа распознаёт жесты рук при помощи веб-камеры по наличию цветных пикселей на изображении. После обнаружения этих пикселей происходит вычисление расстояния между ними. Вычислив это расстояние, программа выдаёт информацию о распознанном жесте. Вид цветных перчаток изображён на рисунке 1.



*Рис. 1. Перчатки, при помощи которых программа распознаёт жесты.*

Этот способ распознавания очень интересен в своей реализации, но он имеет огромный недостаток: необходимость надевания специальных перчаток для того, чтобы программа смогла выделить цветные пиксели. Без них программа неработоспособна.

1.2. Распознавание жестов при помощи сигналов тензодатчиков

Представителем этой группы являются перчатки GRASP – говорящие перчатки для глухонемых (Glove-base Recognition of Auslan using Simple Processing) [9].

Принцип их работы следующий: к компьютеру подключены две перчатки при помощи кабелей. Программа обрабатывает сигналы тензодатчиков и, в зависимости от информации, полученной с этих датчиков, выдаёт информацию о распознанном жесте. Внешний вид устройства приведён на рисунке 2.



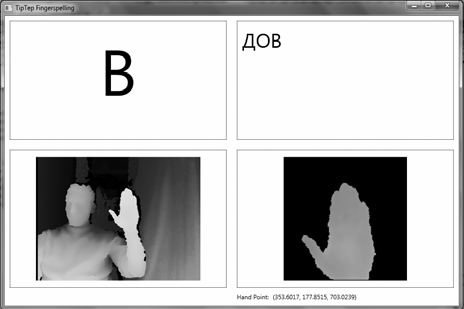
*Рис. 2. Перчатка, при помощи которой программа распознаёт жесты.*

Данный способ реализации так же имеет недостаток в необходимости надевания специальной перчатки, без которой программа не сможет распознать жест.

1.3. Автоматическое преобразование жестов русской ручной азбуки в текстовый вид [1].

Данная программа работает непосредственно с веб-камерой, при помощи которой определяется положение руки в кадре. После строится «скелет» человеческой руки, по которому программа распознаёт жест и переводит его в букву русской азбуки. Вид работы алгоритма приведён на рисунке 3.

Эта программа является наиболее удобной в использовании, так как не требует наличия различных устройств, надетых на руку, но работает только с полутоновыми изображениями, то есть изображениями в оттенках серого. Такие изображения очень светочувствительны, что является сильным недостатком этой программы.



*Рис. 3. Автоматическое преобразование жестов русской ручной азбуки в текстовый вид.*

1.4. Распознавание жестов при помощи нейронной сети

В последнее время нейронные сети получили мощный толчок в своём развитии. Одним из направлений их использования является обработка и извлечения информации из изображений.

Для распознавания жестов искусственная нейронная сеть могут использоваться для её обучения на заранее подготовленных и размеченных изображениях жеста, в таком случае используются свёрточные нейронные сети. Прежде чем начать распознавать жесты на изображении, необходимо локализовать его область, то есть детектировать кисть человека. Для детектирования кисти можно использовать подход Single Shot MultiBox детектора [17].

2. SINGLE SHOT ДЕТЕКТОР

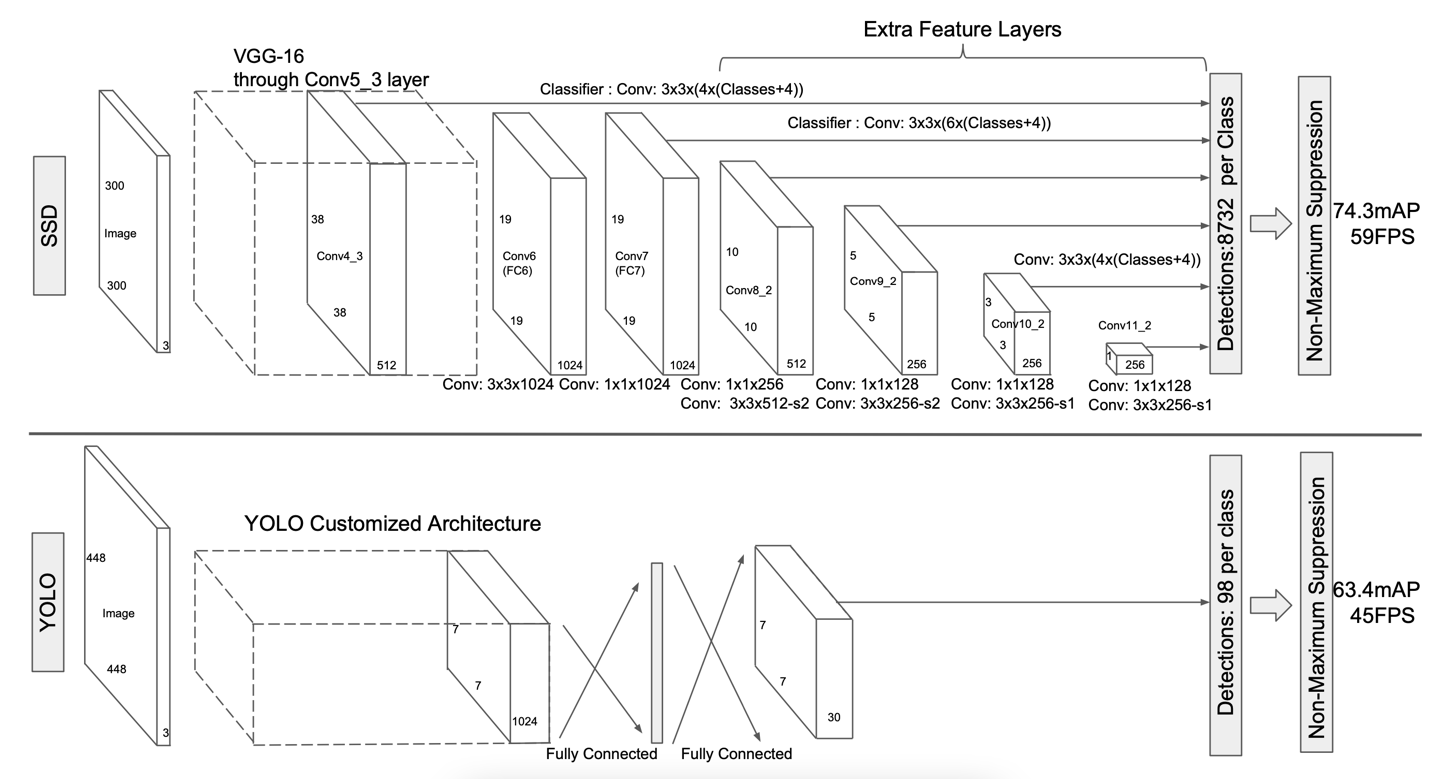
Для работы SSD необходимо только входное изображение и рамки истинности для каждого объекта во время обучения (рис. 4а). С помощью свертки оценивается небольшой набор (например, 4) полей по умолчанию с разными соотношениями сторон в каждом месте не нескольких картах признаков с разными масштабами (например, (рис. 4б) и (рис. 4в)). Для каждой рамки по умолчанию прогнозируется как смещение формы, так и достоверность для всех категорий объектов . Во время обучения сначала сопоставляются эти рамки по умолчанию с рамками истинности. Например, совпали две рамки по умолчанию с котом и одно с собакой на рис. 4а, которые рассматриваются как положительные, а остальные – как отрицательные. Потеря модели представляет собой взвешенную сумму между потерей локализации (Smooth L1 [4]) и потерей достоверности (например, Softmax).

|  |  |  |
| --- | --- | --- |
| а) | б) | в) |
| *Рис. 4. SSD детектор.* | | |

2.1. Модель

Подход SSD основан на свёрточной сети с прямой связью, которая создаёт набор ограничивающих рамок фиксированного размера и оценивает наличие экземпляров класса объектов в этих рамках, с подавлением немаксимумов для получения результатов окончательных обнаружений. Первые слои нейросети основаны на стандартной архитектуре VGG-16, используемой для высококачественной классификации изображений, которая будет называться *базовой сетью*. Затем в сеть добавляется вспомогательная структура для создания обнаружений со следующими ключевыми особенностями:

1. *Мультимасштабные карты ключевых точек для детектирования*. В конец усечённой базовой сети добавляются свёрточные слои ключевых точек. Эти слои постепенно уменьшаются в размере и позволяют прогнозировать обнаружение в нескольких масштабах. Свёрточная модель для прогнозирования обнаружений различается для каждого слоя объектов (Overfeat [15] и YOLO [12]), которые работают с картой признаков в одном масштабе)
2. *Сверточные «предсказатели» для детектирования.* Каждый добавленный слой признаков (или, возможно, существующий слой признаков из базовой сети) может создавать фиксированный набор прогнозов обнаружения с использованием набора свёрточных фильтров. Они указаны поверх сетевой архитектуры SSD на рис. 5. Для слоя признаков размера с каналами базовым элементов для прогнозирования параметров потенциального обнаружения является небольшое ядро , которое даёт либо оценку для категории, либо смещение формы относительно координат рамки по умолчанию. В каждом из положений, где ядро применяется, оно выдает выходное значение. Выходные значения смещения ограничивающей рамки измеряются относительно положения рамки по умолчанию относительно каждого местоположения карты признаков (архитектура YOLO используется промежуточный полносвязный слой вместо свёрточного фильтра для этого шага).



*Рис. 5. Сравнение двух single shot детекторов: SSD и YOLO. В SSD модель в конец базовой сети добавляются несколько ключевых слоёв, которые предсказывают смещения к рамкам по умолчанию различных масштабов и соотношений сторон и связанные с ними достоверности. SSD с входным размером значительно превосходит аналог YOLO по точности и по скорости.*

1. *Рамки и соотношения сторон по умолчанию.* Набор ограничивающих рамок по умолчанию связывается с каждой ячейкой карты признаков для нескольких карт признаков в начале нейросети. Ячейки по умолчанию разбивают карту признаков свёрткой, так что положение каждой ячейки относительно соответствующей ячейки фиксировано. В каждой ячейке карты признаков прогнозируется смещение относительно форм рамок по умолчанию в ячейке, а также оценки для каждого класса, которые указывают на присутствие экземпляра класса в каждой из этих рамок. В частности, для каждой рамки из в заданном месте вычисляются оценки класса и 4 смещения относительно исходной формы рамки по умолчанию. В результате получается фильтров, которые применяются вокруг каждого местоположения на карте признаков, которые дают выходов для карты признаков размера .

2.2. Обучение

Ключевое отличие обучения SSD от обучения типичного детектора, который использует предложения региона (то есть предлагается какой-либо регион изображения и классифицируется) заключается в эталонной информации, которая должна быть назначена конкретным выходам в фиксированном наборе выходов детектора. Некоторая вариация этого также требуется для обучения YOLO [12] и для этапа предложения региона Faster R-CNN [13] и Multibox [2]. Как только это назначение определено, функция потерь и обратное распространение применяются сквозным образом. Обучение также включает в себя выбор набора рамок и масштабов по умолчанию для обнаружения, а также стратегии аугментации и получения полностью отрицательных тренировочных данных (hard negative mining).

1. *Стратегия сопоставления*. Во время обучения необходимо определять, какие рамки по умолчанию соответствуют абсолютно верному обнаружению и тренировать сеть соответствующим образом. Каждая верная рамка, сопоставляются с каждой рамкой, выбранной из рамок по умолчанию, различающихся по местоположению, соотношению сторон и масштабу, с наилучшим коэффициентом Жаккара (как в MultiBox [2]). В отличие от MultiBox, затем в SSD сопоставляются блоки/рамки по умолчанию c истинными блоками с коэффициентом Жаккара выше порогового значения (. Это упрощает задачу обучения, позволяя сети предсказывать высокие значения для нескольких перекрывающихся блоков, а не требовать от неё выбора только одного с максимальным перекрытием.
2. *Цель обучения*. Цель обучения SSD получена из цели Multibox [2], но расширена до обработки нескольких категорий объектов. Пусть индикатор для сопоставления -того блока по умолчанию с -м истинным блоком категории . Из приведённой выше стратегии сопоставления следует, что можно иметь . Общая целевая функция потерь представляет собой взвешенную сумму потерь локализации (loc) и потерь достоверности (conf):

где число совпавших блоков по умолчанию. Если , то потери принимаются равными . Потери локализации – это потери Smooth L1 [4] между предсказанным блоком ( и истинным блоком (). Подобно Faster R-CNN [13] происходит регрессия к смещениям для центра ограничивающего блока по умолчанию (), а также для его ширины () и высоты ().

Потери достоверности – это потери Softmax по достоверности по достоверности нескольких классов ().

и весовой термин устанавливается равным путём перекрёстной проверки.

1. *Выбор масштабов и соотношений сторон для блоков по умолчанию*. Для обработки различных масштабов объектов некоторые методы [15, 7] предполагают обработку изображения разных размеров и последующее объединение результатов. Однако, используя карты признаков из нескольких разных слоёв в одной сети для прогнозирования, можно имитировать тот же эффект, а также использовать общие параметры для всех масштабов объектов. Предыдущие работы [11, 6] показали, что использование карт признаков из нижних слоёв может улучшить качество семантической сегментации, поскольку нижние слои захватывают более мелкие детали входных объектов. Точно так же в [10] показано, что добавление главного контекста, объединённого из карты признаков, может помочь сгладить результаты сегментации. Руководствуясь этими методами, в SSD для обнаружения используются как нижняя, так и верхняя карта признаков.

Известно, что карты признаков с разных уровней внутри сети имеют разные (эмпирические) размеры рецептивного поля [18]. К счастью, в структуре SSD блоки по умолчанию не обязательно должны соответствовать фактическим рецептивным полям каждого слоя. Мозаика блоков по умолчанию разрабатывается таким образом, чтобы определённые карты признаков научились реагировать на определённые масштабы объектов. Предположим, нужно использовать карт признаков для предсказания. Масштаб блоков по умолчанию для каждой карты признаков рассчитывается как:

где , означающие, что самый нижний слой имеет масштаб ,а самый верхний слой и все слои меду ними расположены через равные промежутки. Для блоков по умолчанию устанавливаются различные пропорции и обозначаются как . Можно вычислить ширину () и высоту для каждого блока по умолчанию. Для соотношения сторон также добавляется блок по умолчанию, масштаб которого и в результате получается блоков по умолчанию для каждого местоположения карты признаков. Центр каждого блока по умолчанию устанавливается в , где это размер -ого квадрата карты признаков, . На практике можно также спроектировать распределение блоков по умолчанию, чтобы наилучшим образом соответствовать конкретному набору данных. Вопрос о том, как спроектировать оптимальную мозаику, также остаётся открытым.

Комбинируя прогнозы для всех блоков по умолчанию с разными масштабами и соотношениями сторон из всех местоположений на многих картах признаков, получается разнообразный набор прогнозов, охватывающий различные размеры и формы входных объектов. Например, на рис. 1 собака сопоставляется с блоком по умолчанию на карте признаков , но не с каким-либо блоком по умолчанию на карте признаков . Это связано с тем, что эти блоки имеют разные масштабы и не соответствуют блоку собаки, и поэтому во время тренировки считаются негативными.

1. *Получение полностью отрицательных тренировочных данных (hard negative mining).* После этапа сопоставления большинство блоков по умолчанию являются негативными, особенно когда количество возможных блоков по умолчанию велико. Это вносит значительный дисбаланс между положительными и отрицательными обучающими примерами. Вместо того, чтобы использовать все отрицательные примеры, они сортируются по их наибольшей потере достоверности для каждого блока по умолчанию и выбираются так, чтобы соотношение между отрицательными и положительными примерами было не более чем . Было обнаружено [17], что это приводит к более быстрой оптимизации и более стабильному обучению.
2. *Аугментация данных.* Чтобы сделать модель более устойчивой к различным размерам и формам входных объектов, каждое обучающее изображение случайным образом выбирается одним из следующих параметров:
   * Использовать всё исходное входное изображение.
   * Выбрать фрагмент так, чтобы минимальный коэффициент Жаккара с объектами составлял или .
   * Выбрать фрагмент произвольно.

Размер каждого фрагмента выборки составляет от исходного размера изображения, а соотношение сторон находится в диапазоне от до . Перекрывающаяся часть истинного блока сохраняется, если его центр находится в фрагменте выборки. После вышеупомянутого шага размер каждого фрагмента изменяется до фиксированного и фрагмент горизонтально отражается с вероятностью , в дополнение к применению некоторых фотометрических искажений, подобных описанным в [8].

2.3. Аналоги

Существует два класса методов детектирования объектов на изображениях: один основан на скользящих окнах, а другой – на классификации предложений регионов. До появления свёрточных нейронных сетей уровень развития этих двух подходов – модели DPM (Deformable Part Model [3]) и выборочного поиска [16] – имели сопоставимую производительность. Однако после резкого улучшения, внесённого R-CNN (Region-Based Convolutional Neural Network) [5], который сочетает с себе выборочный поиск предложений регионов и пост-классификацию на основе свёрточной сети, методы обнаружения объектов на основе предложений регионов стали преобладающими. Сравнение подходов представлено в таблице 1. SSD300 – единственный метод, способный работать в реальном времени и набравший более 70% mAP (mean average precision – средняя точность предсказания), являющийся компромиссом между скоростью и качеством работы.

Таблица 1.

Сравнение подходов на тесте Pascal VOC2007.

|  |  |  |
| --- | --- | --- |
| Метод | mAP, % | FPS |
| Faster R-CNN (VGG16) |  |  |
| Fast YOLO |  |  |
| YOLO (VGG16) |  |  |
| SSD300 |  |  |
| SSD512 |  |  |

Первоначальный подход R-CNN был улучшен различными способами. Первый набор подходов улучшает качество и скорость пост-классификации, поскольку требует классификации тысяч кадров изображений, что дорого и затратно по времени. SPPnet [7] значительно ускоряет исходный подход R-CNN за счёт ввода слоя объединения слоя объединения пространственных пирамид, который более устойчив к размеры и масштабу региона и позволяет слоям классификации переиспользовать признаки, вычисленные по картам признаков, сгенерированным для нескольких разрешений изображения. Fast R-CNN [4] расширяет возможности SPPnet, так что он может точно настраивать все уровни от начала до конца, минимизируя потери как для достоверности, так и для регрессии ограничивающей рамки, которая впервые была представлена в MultiBox [6] для изучения объектности.

Второй набор подходов улучшает качество генерации предложений, используя глубокие нейронные сети. В самых последних работах, таких как MultiBox [2], предложения региона выборочного поиска, которые основаны на низкоуровневых признаках изображения, заменены предложениями, сгенерированными непосредственно из отдельной глубокой нейронной сети. Это дополнительно повышает точность обнаружения, но приводит к несколько сложной настройке, требующей обучения двух нейронных сетей с зависимостью между ними. Faster R-CNN [13] заменяет предложения выборочного поиска предложениями, полученными из сети региональных предложений (RPN), и вводит метод интеграции RPN с Fast R-CNN путем чередования точной настройки общих свёрточных слоёв и слоёв прогнозирования для этих двух сетей. Таким образом, предложения регионов используются для объединения признаков среднего уровня, а последний этап классификации обходится дешевле. SSD очень похож на сеть региональных предложений (RPN) в Faster R-CNN, поскольку также используется фиксированный набор блоков (по умолчанию) для прогнозирования, аналогично полям привязки в RPN. Но вместо того, чтобы использовать их для объединения признаков и оценки другого классификатора, оценка для каждой категории объектов получается одновременно в каждом блоке. Таким образом, этот подход позволяет избежать осложнений, связанных с объединением RPN с Fast R-CNN, и его легче обучать, быстрее и проще интегрировать в другие задачи.

Другой набор методов, непосредственно связанных с подходом SSD, полностью пропускает шаг предложения и напрямую прогнозирует ограничивающие рамки и достоверность для нескольких категорий. OverFeat [15], глубокая версия метода скользящего окна, предсказывает ограничивающую рамку непосредственно из каждого местоположения самой верхней карты признаков после того, как известны достоверности базовых категорий объектов. YOLO [12] использует всю самую верхнюю карту признаков для прогнозирования как достоверности для нескольких категорий, так и ограничивающих рамок (которые являются общими для этих категорий). Метод SSD попадает в эту категорию, поскольку в нём нет шага предложения, но используются блоки по умолчанию. Однако этот подход является более гибким, чем существующие методы, поскольку существует возможность использовать блоки по умолчанию с разными соотношениями сторон для каждого местоположения признака из нескольких карт признаков в разных масштабах. Если будет использоваться только один блок по умолчанию для каждого местоположения из самой верхней карты признаков, то SSD будет иметь архитектуру, аналогичную OverFeat [15]; если используется вся самая верхняя карта признаков и добавляется полносвязный слой для прогнозов вместо свёрточных предсказателей и не будет явно учитываться несколько соотношений сторон, то приблизительно воспроизведётся архитектура YOLO [12, 2].

2.4. Выводы

Таким образом, резюмируя всё выше сказанное, можно отметить следующие особенности Single Shot детектора:

1. Архитектура SSD позволяет детектировать объекты в реальном времени;
2. Качество работы близко к Faster R-CNN;
3. Детектирование происходит на разных масштабах, что позволяет локализовывать объекты разных размеров;
4. Используется большое количество рамок по умолчанию, покрывающих входное изображение на разных масштабах;
5. На этапе работы архитектура SSD300 осуществляет детектирование 7308 объектов, большая часть из которых впоследствии фильтруется;

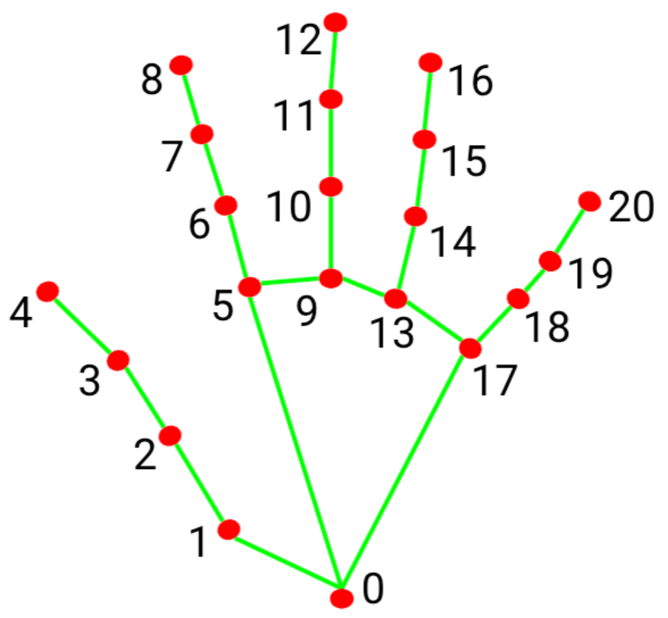
Ключевой же особенностью является использование многомасштабных ограничительных рамок на выходах свёрточных слоёв, прикреплённых к нескольким картам признаков в верхней части сети. Экспериментально подтверждено [17], что при соответствующих стратегиях обучения, большее количество тщательно выбранных ограничивающих рамок по умолчанию приводит к повышению производительности.

Для решения задачи поиска ладони Single Shot детектор подходит наилучшим образом, поскольку он имеет хорошее соотношение между скоростью работы и точностью. Этот же детектор также используется в библиотеке MediaPipe Hands, использующейся для поиска ключевых точек на руках. Рассмотрим её поподробнее.

2.5. Библиотека MediaPipe Hands

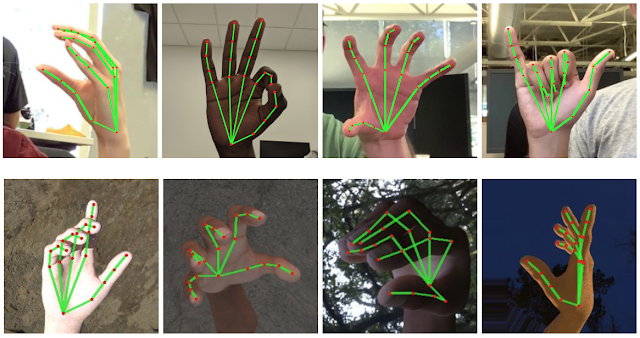
Для определения начального положения рук была разработана модель SSD, оптимизированная для использования в реальном времени. Поскольку обнаружение рук чрезвычайно сложная задача из-за огромного количества их конфигураций, то вместо детектора рук обучался детектор ладоней, так как получать ограничивающие рамки негнущихся объектов, таких как ладони и кулаки, значительно проще, нежели обнаружение рук вместе с пальцами. Кроме того, поскольку ладони являются более мелкими объектами, алгоритм подавления немаксимумов хорошо работает даже для случаев пересечения двух рук, таких как рукопожатия. Более того, ладони могут быть смоделированы с помощью квадратных ограничивающих рамок (якорей), игнорируя другие соотношения сторон, уменьшая количество якорей в 3-5 раз. Во-вторых, кодирующий-декодирующий экстрактор признаков используется для большей осведомленности о контексте сцены даже для небольших объектов. Наконец, во время обучения минимизируется функция кросс-энтропийных потерь с динамическим масштабированием (focal loss), чтобы поддерживать большое количество якорей, возникающих из-за большой дисперсии масштаба. С помощью описанных выше методов достигается средняя точность обнаружения ладони . Использование обычной функции кросс-энтропийных потерь и без декодера даёт точность всего .

После обнаружения ладони на всём изображении последующая модель выполняет точную локализацию 21 ключевой точки (рис. 6) внутри обнаруженных областей руки посредством регрессии (как для определения цифры, написанной рукой), то есть прямого прогнозирования координат. Модель обучается последовательному внутреннему представлению позиции рук и устойчива даже к частично видимым рукам и самопересечениям.



*Рис. 6. Ключевые точки на руке, определяемые с помощью MediaPipe Hands.*

Чтобы получить данные для обучения, вручную были промаркировано около изображений реального мира с 21 координатой, как показано ниже на рис. 7. А также чтобы лучше охватить возможные позиции рук и обеспечить дополнительное наблюдение за характером геометрии рук была визуализирована высококачественная синтетическая модель руки на различных фонах.



*Рис. 7. Сверху: реальные изображения рук, аннотированные вручную. Снизу: сгенерированные синтетические изображения рук с аннотациями.*

Таким образом, библиотека состоит из детектора ладоней, определяющего область изображения, на которой изображена рука и модели трекинга руки, предсказывающая ключевые точки на области, полученной детектором.

Основываясь на данной технологии, полученные ключевые точки можно использовать в дальнейших приложениях, например, в управлении курсором мыши.

3. МЕТОДИКА УПРАВЛЕНИЯ КУРСОРОМ МЫШИ

Управление курсором мыши с помощью жестов будет осуществляться на основе ключевых точек, полученных с помощью MediaPipe Hands. Поскольку с помощью жестов управления курсором можно имитировать жесты трекпада, которые часто имеют разную конфигурацию в зависимости от производителя, то для гибкости использования методики следует реализовать два режима работы: «*обучение»*, которое будет включать в себя добавление новых жестов, и «*применение»*, идентифицирующее жест и выполняющее конкретное заранее назначенное действие для него.

3.1. Сбор данных

Для быстрого добавления новых жестов в существующую систему необходимо учесть следующие требования:

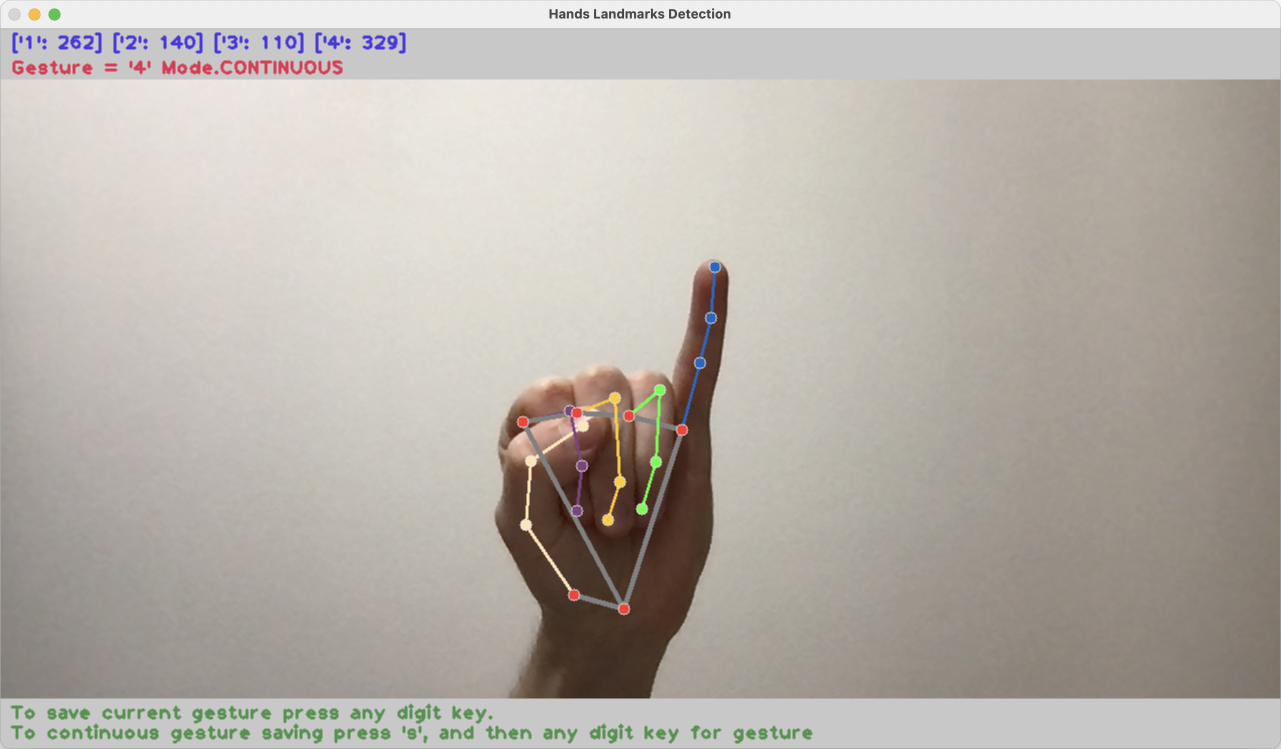
* Выбор режима работы перед запуском программы;
* Переключение между одной и последовательной записью жестов;
* Онлайн-мониторинг количества уже записанных координат по каждому из жестов;
* Вывод текущего записанного жеста;

Для удовлетворения данных требований был разработан интерфейс, представленный на рис. 8.



*Рис. 8. Интерфейс программы с пометками.*

Запись жестов осуществляется при нажатии на одну из цифровых клавиш ‘0’-‘9’ (по номеру записываемого жеста). При нажатии на клавишу ‘s’ активируется режим последовательной записи жеста, то есть координаты сохраняются непрерывно, а не только при нажатии на цифру (рис. 9). Данный подход позволяет быстрее собирать большое количество данных, непрерывно перемещая руку с жестом. В верхней части интерфейса выводится количество записей по каждому из жестов, предоставляя возможность выравнивать количество данных по ним. Для выхода из программы достаточно нажать ‘q’ или ESC.

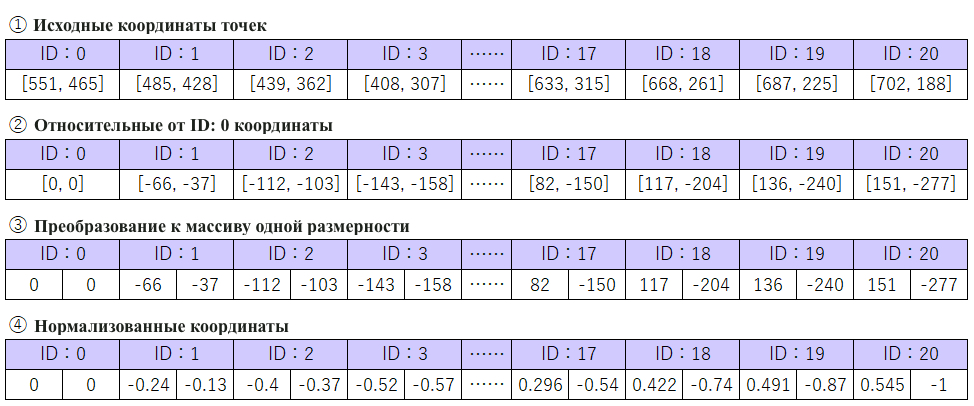


*Рис. 9. Непрерывная запись четвёртого жеста.*

Собрав необходимое количество данных, необходимо обучить модель распознавать конкретный жест исходя из ключевых точек.

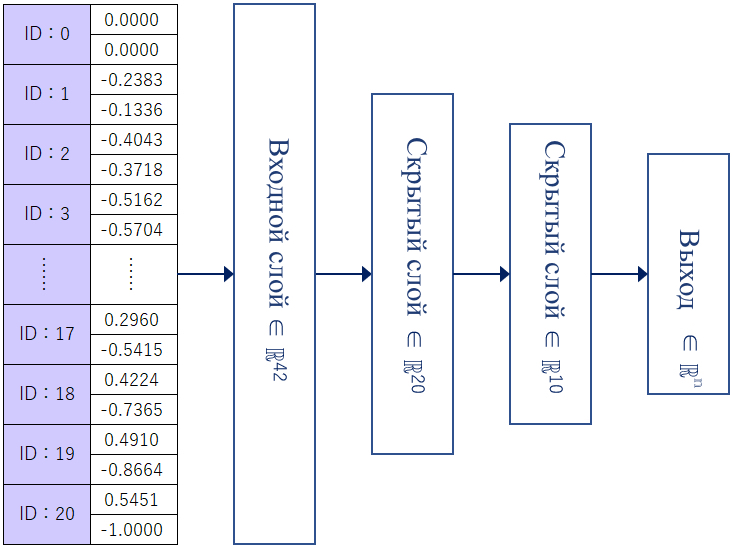
3.2. Обучение модели

Подготовка данных для обучения модели заключается в преобразовании координат к необходимому нормализованному виду, как, например, на рис. 10. Исходные значения координат поступают с привязкой к расположению ключевой точки на изображении, то есть в пикселях. Поскольку жест не всегда может находится в конкретном данном расположении, то необходимо отвязать их от изображения. Для этого все координаты пересчитываются относительно ключевой точки, находящейся на запястье, и нормируются.



*Рис. 10. Преобразование от исходных координат к нормализованным.*

Затем, набор массивов координат, привязанных к конкретным жестам, подаётся на вход искусственной нейронной сети с предварительно отобранной структурой, показанной на рис. 11.



*Рис. 11. Структура нейронной сети. Размер выхода зависит от количества жестов.*

Более подробно структуру можно записать следующим образом:

* Входной слой
* Дропаут 20%
* Скрытый слой () с функцией активации ReLU:
* Дропаут 40%
* Скрытый слой () с функцией активации ReLU
* Выходной слой () с функцией активации Softmax, где количество жестов

*Исключение* или *дропаут* – метод регуляризации искусственных нейронных сетей, предназначен для уменьшения переобучения сети за счёт предотвращения сложных коадаптаций отдельных нейронов на тренировочных данных во время обучения, характеризует исключение определённого процента случайных нейронов на разных итерациях во время обучения нейронной сети. Такой приём значительно увеличивает скорость обучения, качество обучения на тренировочных данных, а также повышает качество предсказаний модели на новых тестовых данных.

В качестве примера были выбраны 3 жеста с одним, двумя и тремя пальцами соответственно и заранее сгенерированный набор из 3138 точек для этих жестов. Обучение для него проходило в 148 эпохах и составило всего 10 секунд. Доля правильных ответов (accuracy) , потери (loss) . Основные метрики по каждому классу представлены в таблице 2.

Таблица 2.

Показатели метрик классификации.

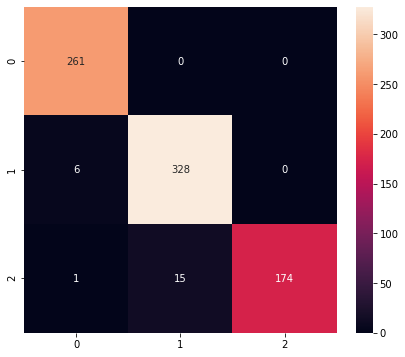
|  |  |  |  |
| --- | --- | --- | --- |
| Класс | Точность (precision) | Полнота (recall) | F-мера (f1-score) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Рассмотрим каждую из метрик. Перед переход к самим метрикам необходимо ввести важную концепцию для описания метрик в терминах ошибок классификации – confusion matrix (матрица ошибок). Допустим, что у нас есть два класса и алгоритм, предсказывающий принадлежность каждого объекта к одному из классов, тогда матрица ошибок классификации будет выглядеть следующим образом:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | True Positive (TP) | False Positive (FP) |
|  | False Negative (FN) | True Negative (TN) |

*Таблица 3. Матрица ошибок для бинарной классификации.*

Здесь   это ответ алгоритма на объекте, а истинная метка класса на этом объекте. Таким образом, ошибки классификации бывают двух видов: False Negative (FN) и False Positive (FP). Матрица ошибок для задачи распознавания жестов представлена на рис. 12.



*Рис. 12. Матрица ошибок для задачи распознавания жестов.*

Интуитивно понятной, очевидной и почти неиспользуемой метрикой является accuracy – доля правильных ответов алгоритма:

Эта метрика бесполезна в задачах с неравными классами, покажем это на примере. Допустим, мы хотим оценить работу спам-фильтра почты. У нас есть 100 не-спам писем, 90 из которых на классификатор определил верно (True Negative = 90, False Positive = 10), и 10 спам-писем, 5 из которых классификатор также определил верно (). Тогда

Однако, если просто будем предсказывать все письму как не-спам, то получим более высокую accuracy:

Для оценки качества работы алгоритма на каждом из классов по отдельности введём метрики precision (точность) и recall (полнота):

Precision можно интерпретировать как долю объектов, названных классификатором положительными и при это действительно являющимися положительными, а recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашёл алгоритм. Существует несколько различных способов объединить precision и recall в агрегированный критерий качества. F-мера (в общем случае ) – среднее гармоническое precision и recall:

в данном случае определяет вес точности в метрике, и при это среднее гармоническое. F-мера достигает максимума при полноте и точности, равными единице, и близка к нулю, если один из аргументов близок к нулю.

Проанализировав результаты, можно сделать вывод, что модель успешно обучена и её можно применить в работе.

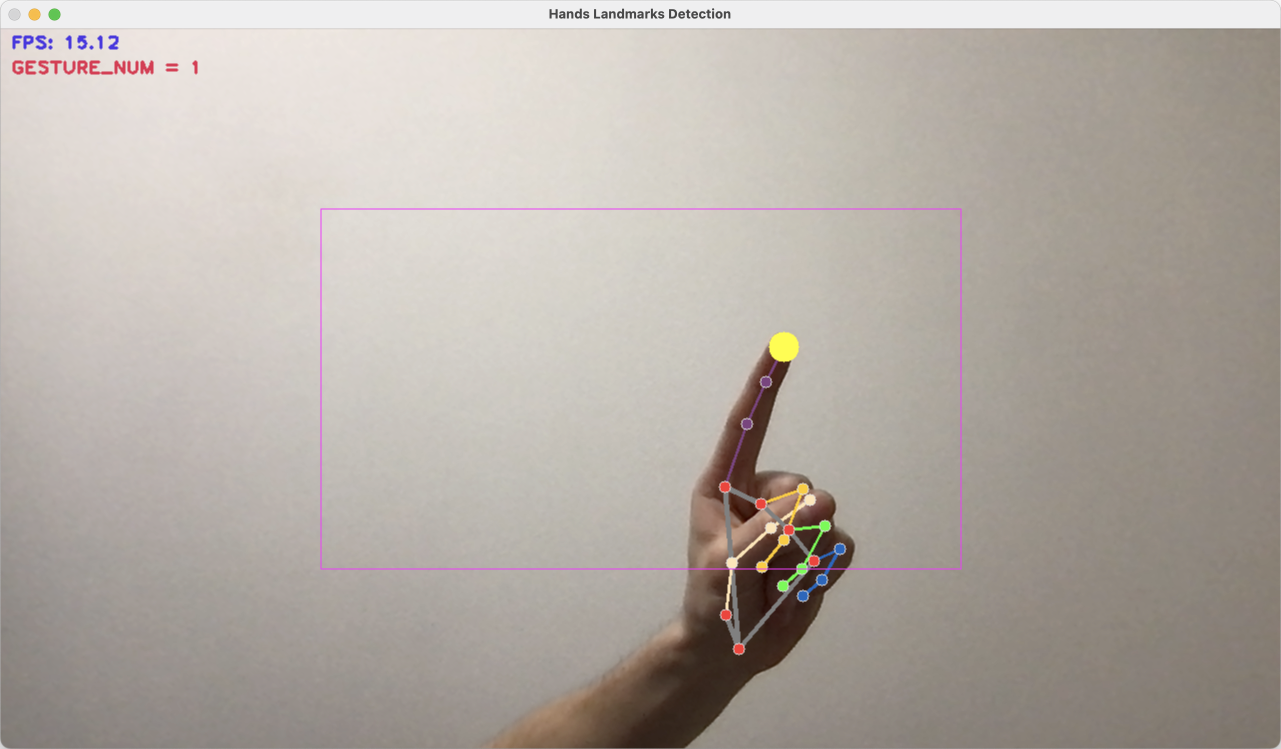
3.3. Практическое применение

В качестве практического применения распознавания жеста было выбрано управление курсором мыши. В качестве жестов были выбраны следующие конфигурации:

* Жест 1 – показывается только указательный палец, остальные собраны в кулак;
* Жест 2 – по аналогии с 1 показывается только указательный и средний палец;
* Жест 3 – по аналогии с предыдущими показывается указательный, средний и безымянный пальцы;

Пусть жест 1 обозначает перемещение курсора, где верхняя точка указательного пальца обозначает сам курсор, а жест 2 и 3 – нажатие левой и правой кнопки мыши соответственно. Интерфейс программы изображён на рис. 13. В верхней части отображается количество кадров в секунду (FPS, Frames Per Second) и номер распознанного жеста. В центральной части отображается область проекции экрана, то есть левый верхний угол данной области соответствует левому верхнему углу экрана и так далее. Перемещая указательный палей в данной области, перемещается курсор по экрану. Причина, по которой была выбрана именно область, а не всё изображение, состоит в том, что в крайних частях изображения плохо или вовсе не распознается жест, поскольку рука не помещается в кадр. Для разных экранов область можно масштабировать и перемещать с помощью следующих клавиш:

* ‘,’/’.’ (‘б’/’ю’ в русской раскладке) – уменьшение/увеличение области соответственно;
* ‘j’/’k’/’l’/’;’ (‘о’/’л’/’д’/’ж’) – сдвиг области влево/вправо/вверх/вниз соответственно;



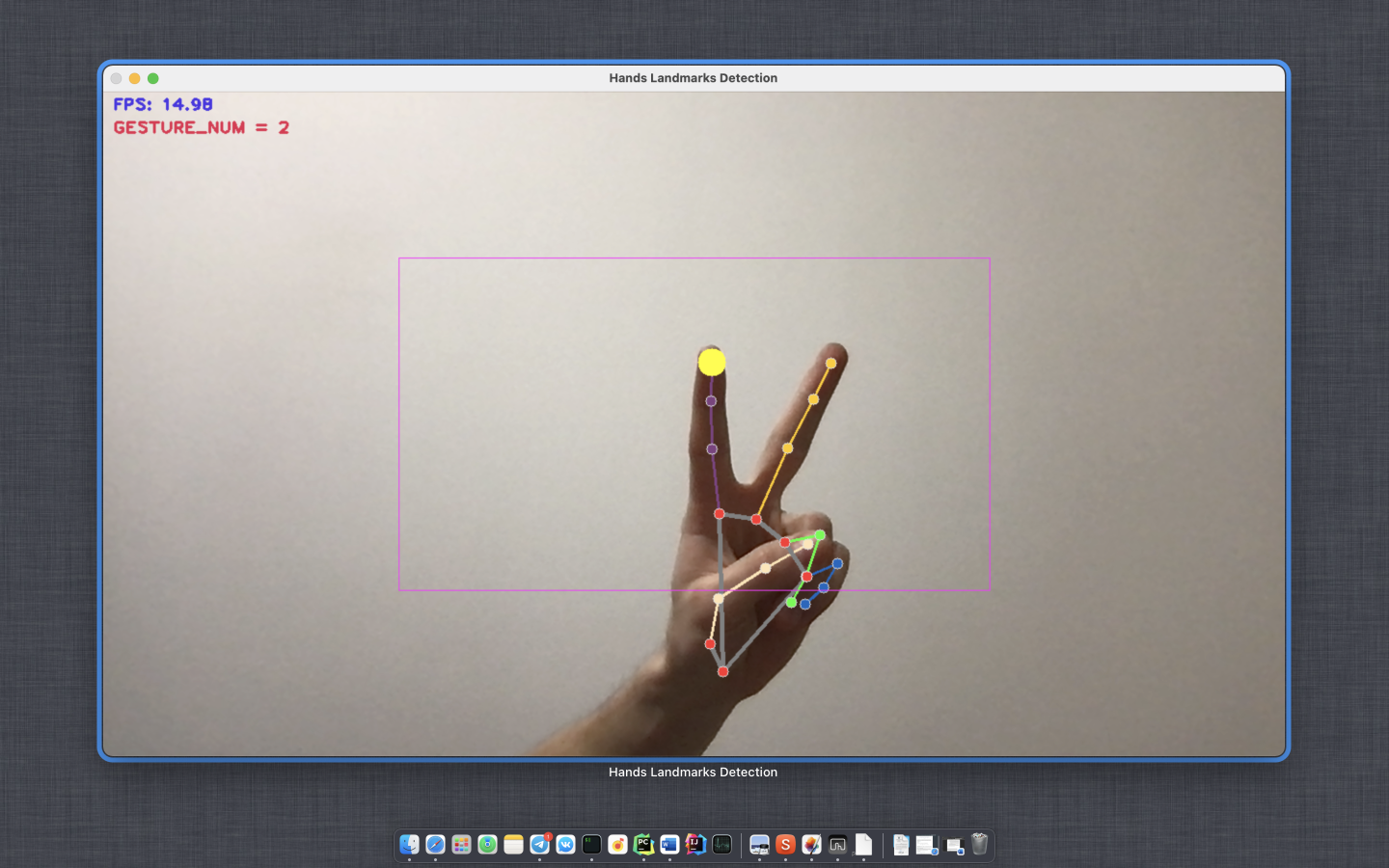
*Рис. 13. Интерфейс программы при управлении курсором.*

При первом тестировании такой системы был найден недостаток такого подхода – из-за небольшой тряски руки курсор постоянно находится в движении и трудно корректно его позиционировать. В качестве решения был предложен метод *фильтр перемещения в малом*, заключающийся в игнорировании малых перемещений точки в пределе пикселей, решивший данную проблему.

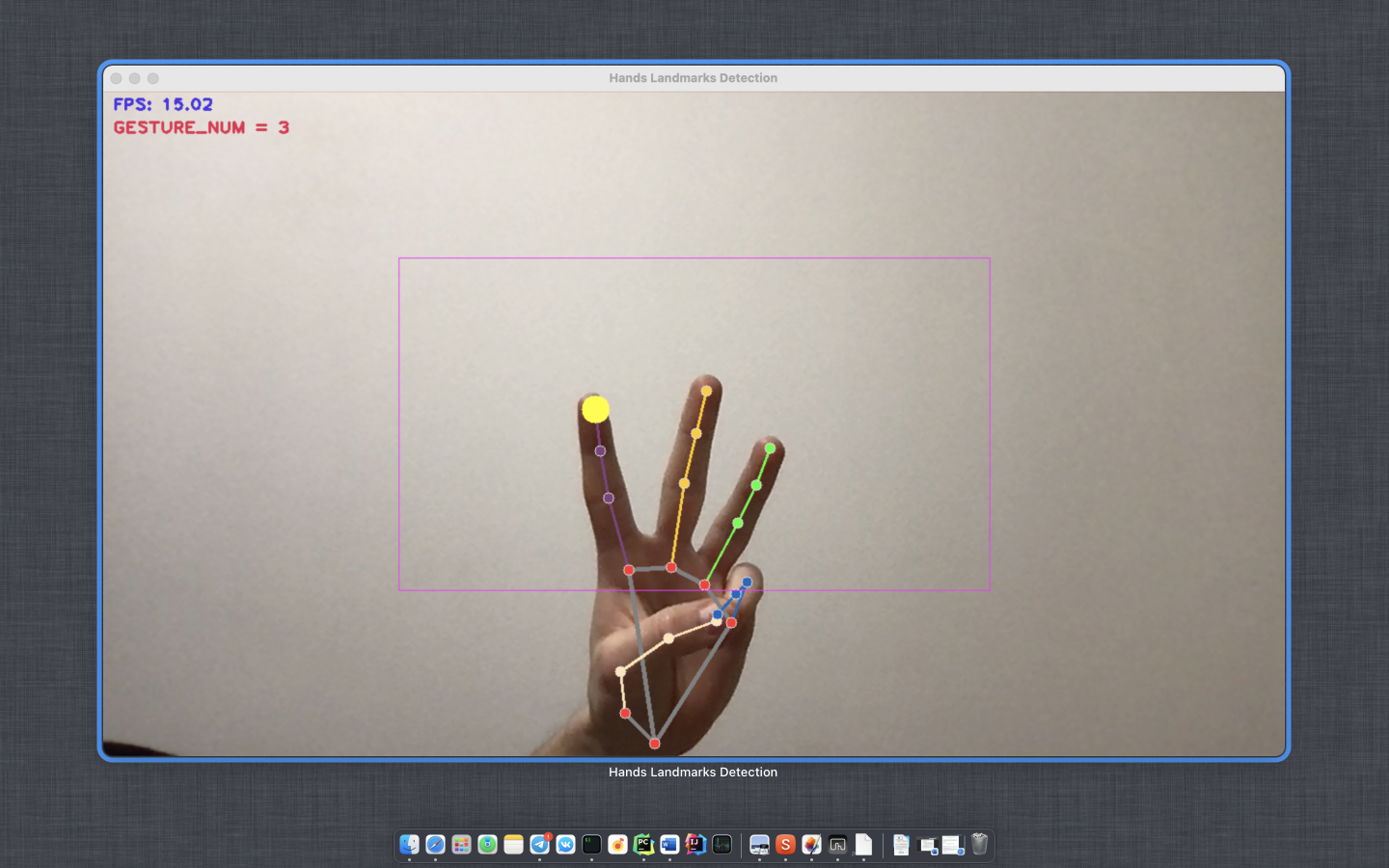
При последующих тестированиях алгоритм работал достаточно точно и быстро, в большинстве случаев верно определяя ключевые точки и классифицируя их, выдавая заданный жест. Неточности возникали при больших наклонах руки – это связано прежде всего с неудачными данными для обучения. При увеличении количества различных ситуаций и количества данных алгоритм работал еще точнее.

В качестве *основного недостатка* данной системы можно выделить неустойчивость при резких передвижениях руки в кадре. Это вызвано тем, что алгоритм предсказывает следующее положение руки в малой окрестности, а при резком перемещении срабатывает детектор, которому необходимо немного больше времени для работы распознавания.

Примеры распознавания второго и третьего жеста показаны на рис. 14 и 15.



*Рис. 14. Экран с запущенной программой, распознающей второй жест.*



*Рис. 15. Экран с запущенной программой, распознающей третий жест.*

ЗАКЛЮЧЕНИЕ

Детектирование рук и распознавание жестов являются одними из наиболее популярных прикладных задач компьютерного зрения, но до сих пор в их решении имеются проблемы с получением точного результата из-за огромного количество конфигураций кисти.

В результате данной работы были рассмотрены уже существующие решения на основе самых разнообразных алгоритмов и способов, но большинство из них работают с использованием каких-либо других инструментов, что не является удобным для конечного потребителя.

Затем были подробно изучены теоретические аспекты актуального способа детектирования объектов на основе искусственных нейронных сетей – Single Shot детектора. В результате сравнения с аналогами был сделан вывод, что данный детектор наилучшим образом подходит для решения требуемой задачи. Также была описана работа современной библиотеки MediaPipe Hands, использующей Single Shot детектор и регрессию для определения 21 ключевой точки на руке.

После этого на основе библиотеки была разработана методика обучения и распознавания жестов для управления курсором мыши. Для этого:

* был собран набор данных, состоящий из 3000 записей ключевых точек для трёх типов жестов;
* спроектирована и обучена искусственная нейронная сеть, классифицирующая данные жесты со средней точностью по классам равной ;
* протестирован на видеопоследовательности реализующий методику алгоритм и исправлены основные недостатки;

По итогу работы следует отметить, что точности и трёх типов жестов не всегда достаточно для управления особо чувствительными системами. Поэтому в методику был включен режим обучения, позволяющий легко расширить собранный набор данных новыми или уже существующими жестами.

В дальнейшем есть возможность проектирования интерфейса для управления не только курсором мыши, но и любыми системами, использующими человеко-машинный интерфейс.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. В.Э. Нагапетян, В.М. Хачумов. Автоматическое преобразование жестов русской ручной азбуки в текстовый вид // Искусственный интеллект и принятие решений. 2013. с. 59-66.
2. Erhan, D., Szegedy, C., Toshev, A., Anguelov, D. Scalable object detection using deep neural networks. CVPR. 2014.
3. Felzenszwalb, P., McAllester, D., Ramanan, D. A discriminatively trained, multiscale, deformable part model. CVPR. 2008.
4. Girshick, R. Fast R-CNN. ICCV. 2015.
5. Girshick, R., Donahue, J., Darrell, T., Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR. 2014.
6. Hariharan, B., Arbelaez, P., Girshick, R., Malik, J. Hypercolumns for object segmentation and fine-grained localization. CVPR. 2015.
7. He, K., Zhang, X., Ren, S., Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. ECCV. 2014.
8. Howard, A.G. Some improvements on deep convolutional neural network based image classification. 2013.
9. Kadous, Mohammed. Auslan Sign Recognition Using Computers and Gloves. 1998.
10. Liu W., Rabinovich A., Berg A.C. ParseNet: Looking wider to see better. ICLR. 2016.
11. Long, J., Shelhamer, E., Darrell, T. Fully convolutional networks for semantic segmentation. CVPR. 2015.
12. Redmon, J., Divvala, S., Girshick, R., Farhadi, A. You only look once: Unified, real-time object detection. CVPR. 2016.
13. Ren, S., He, K., Girshick, R., Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. NIPS. 2015.
14. Robert Y. Wang and Jovan Popovic. Real-Time Hand-Tracking with a Color Glove. ACM Transaction on Graphics. 2009.
15. Sermanet P., Eigen D., Zhang X., Mathieu M., Fergus R., LeCun Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. ICLR, 2014.
16. Uijlings, J.R., van de Sande, K.E., Gevers, T., Smeulders, A.W. Selective search for object recognition. IJCV. 2013.
17. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. SSD: Single Shot MultiBox Detector. 2016.
18. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A. Object detectors emerge in deep scene cnns. ICLR. 2015.