Министерство образования Республики Беларусь

Учреждение образования
ГРОДНЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ ЯНКИ КУПАЛЫ

Физико-технический факультет
Кафедра информационных систем и технологий

К защите допустить:
Заведующий кафедрой ИСиТ
_____Ю.Р.Бейтюк
«____»_____20__г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
на тему

РЕАЛИЗАЦИЯ СИСТЕМЫ HMI ДЛЯ БЕСКОНТАКТНОГО УПРАВЛЕНИЯ ОБОРУДОВАНИЕМ НА ОСНОВЕ ЖЕСТОВЫХ КОМАНД

ГРГУ ДП 1-38 02 01 08 006 ПЗ

Студент	_____	М.Р. Ступакевич
Руководитель	_____	Б.А. Ассанович
Нормоконтролер	_____	Ю.Р. Бейтюк
Рецензент	_____	

Гродно 2022

РЕФЕРАТ

Дипломный проект «Реализация системы HMI для бесконтактного управления оборудованием на основе жестовых команд» содержит 104 страницы, 31 изображение, 2 таблицы, 14 источников литературы.

Цель данной работы: реализация ПО идентификации жестовых команд в видеопотоке, поступающем с камеры.

Методы исследования: для реализации проекта использовались текстовый редактор Visual Studio Code и Jupyter Notebook, язык программирования Python.

Полученные результаты: приложение сбора данных для обучения нейронной сети и приложение системы HMI для бесконтактного управления оборудованием.

Область применения: проект может быть использован в системах требующих бесконтактного управления.

RESUME

The diploma project «Realisation the HMI system for contactless control of equipment based on gesture commands» contains 104 pages, 31 images, 2 tables, 14 literature sources.

The purpose of this work is to realize the identification of gesture commands in the video stream coming from the camera.

Research methods: Visual Studio Code text editor and Jupiter Notebook, Python programming language were used to realize the project.

The results obtained are: a data acquisition application for training a neural network and an HMI system application for contactless control of equipment.

Scope: the project can be used in systems requiring contactless control.

СОДЕРЖАНИЕ

звонок, «Стоп» для паузы, «Лайк» и «Дизлайк» для оценки видеоконтента и «Тихо» для включения/выключения микрофона в звонке.

Рисунок 1 - Жесты SberBox

Одним из последних обновлений программного обеспечения для SberBox Top стали возможность распознавания физических активностей для работы с устройством в качестве фитнес помощника.

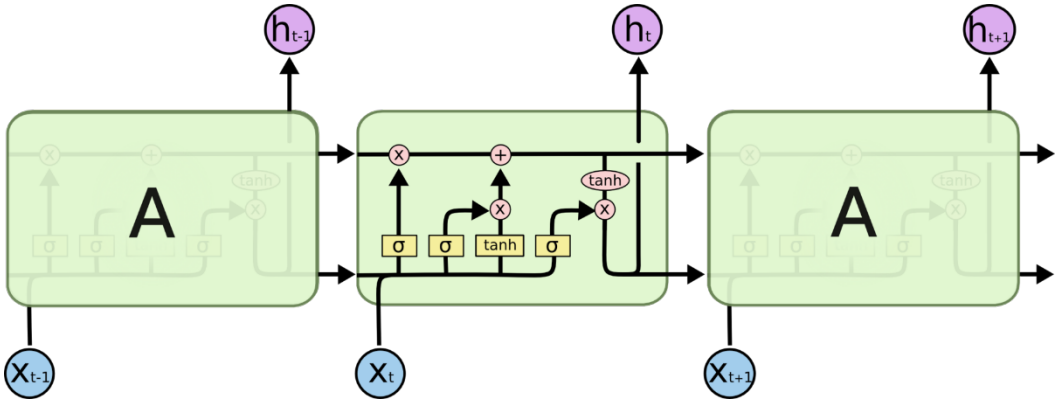


Рисунок 3 - Структурная схема LSTM слоя

Ключевой компонент LSTM – это состояние ячейки (cell state) – горизонтальная линия, проходящая по верхней части схемы (Рисунок - 4).

Состояние ячейки напоминает конвейерную ленту. Он проходит непосредственно через всю цепочку, участвуя лишь в нескольких линейных преобразованиях. Информация может легко проходить через него без изменений.

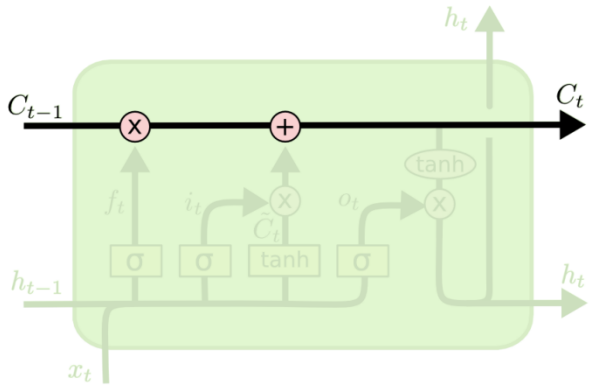


Рисунок 4 - Состояние ячейки

Тем не менее, LSTM может удалять информацию из состояния ячейки; этот процесс регулируется структурами, называемыми фильтрами (gates).

Фильтры позволяют пропускать информацию на основании некоторых условий. Они состоят из слоя сигмоидальной нейронной сети и

операции поточечного умножения (Рисукок – 5 Сигмоидальный слой).

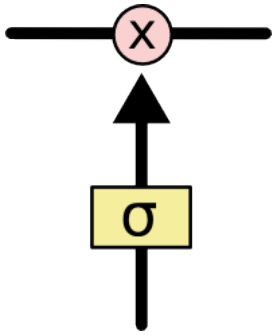
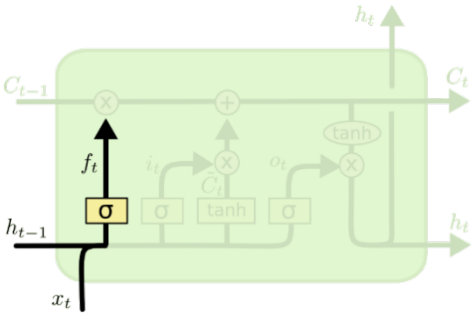


Рисунок 5 - Сигмоидальный слой

Сигмоидальный слой возвращает числа от нуля до единицы, указывающие, какая часть каждого блока информации должна пройти дальше по сети. Ноль в данном случае означает "ничего не пропускать", единица - "пропускать все".

В LSTM три таких фильтра, позволяющих защищать и контролировать состояние ячейки.

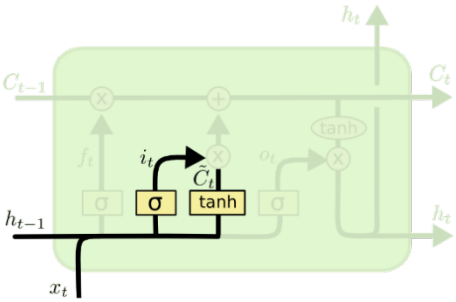
Первый шаг в LSTM – определить, какую информацию можно выбросить из состояния ячейки. Это решение принимает сигмоидальный слой, называемый “слоем фильтра забывания” (forget gate layer) (Рисунок – 6). Он смотрит на h_t-1 и x_t и возвращает число от 0 до 1 для каждого числа из состояния ячейки C_{t-1}. 1 означает “полностью сохранить”, а 0 – “полностью выбросить”.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Рисунок 6 - Слой фильтра забывания

Следующий шаг — решить, какая новая информация будет храниться в состоянии ячейки. Этот этап состоит из двух частей. Во-первых, сигмоидальный слой, называемый «воротом входного слоя», определяет, какие значения необходимо обновить (Рисунок – 7). Затем tanh-слой строит вектор новых значений-кандидатов C_t, которые можно добавить в состояние ячейки.

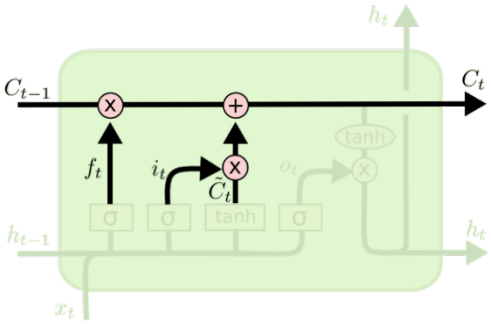


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Рисунок 7 - Слой входного фильтра

Настало время заменить старое состояние ячейки C_{t-1} на новое состояние C_t.

Необходимо умножить старое состояние на f_t, забывая то, что решено забыть. Затем прибавляется $i_t * C_t$. Это новые значения-кандидаты, умноженные на t – на сколько следует обновить каждое из значений состояния (Рисунок - 8).



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Рисунок 8 - Слой обновления состояния

3 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

3.1 Общая структура

Конечное приложение, разработанное в этом дипломном проекте, должно работать согласно приведённой ниже структурной схеме:

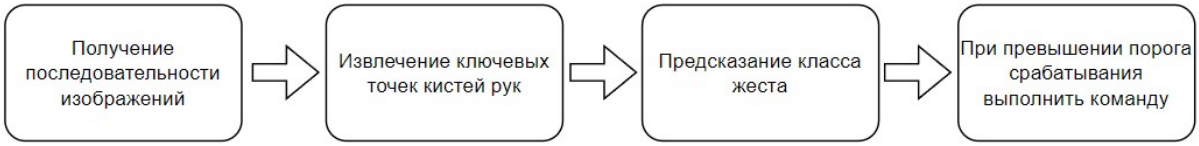


Рисунок 9 - Структурная схема системы

Основным действующим элементом HMI системы является модель рекуррентной нейронной сети LSTM, выполняющая функцию машинного обучения с учителем, а именно - классификацию. Для использования её сначала нужно обучить на реальных размеченных данных. Таким образом, формулируются следующие промежуточные задачи: реализация инструмента сбора и мгновенной разметки данных, формулирование жестовых команд, которые система должна будет распознать, сбор данных, обработка собранных данных.

Для решения первой задачи разработано приложение, работающее по представленной ниже структурной схеме:

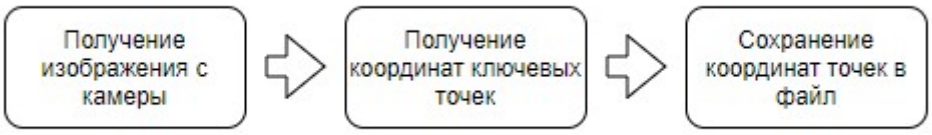


Рисунок 10 - Структурная схема работы приложения сбора данных

Далее, согласно сформулированным для классификации жестовым командам, следует собрать данные и произвести их обработку

(визуализация, анализ, интерполяция и другие преобразования), что будет выполнено согласно следующей схеме:

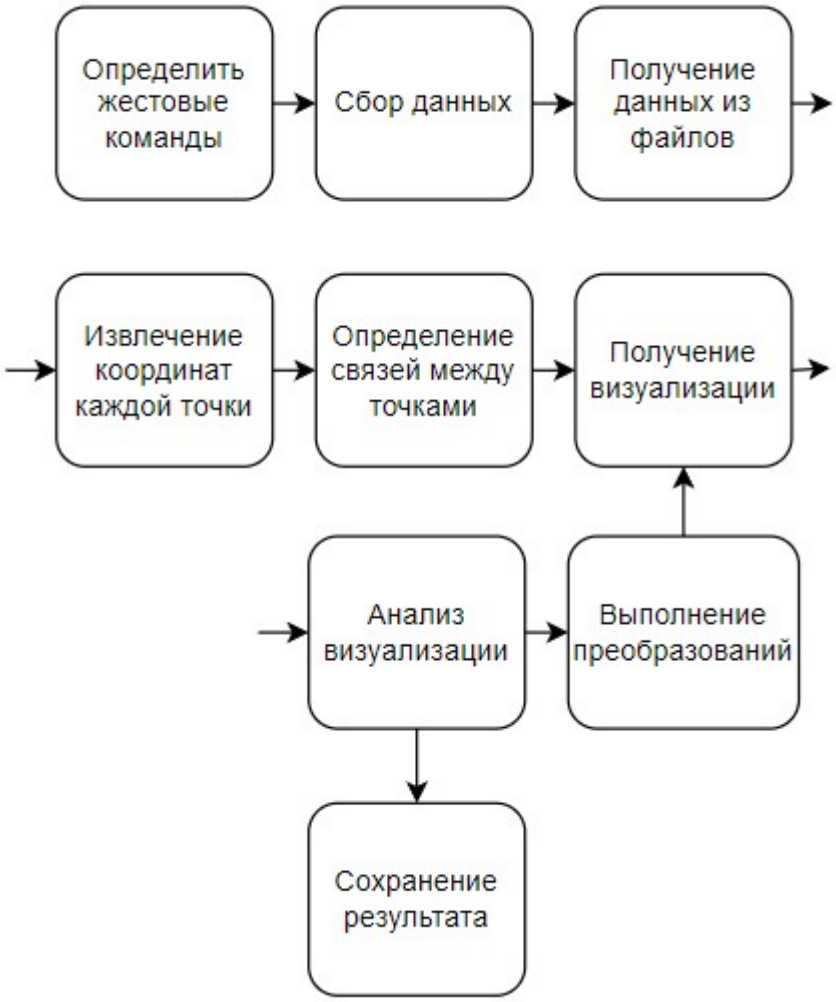


Рисунок 11 - Структурная схема этапов сбора и обработки данных

Таким образом, мы получаем всё необходимое для обучения модели рекуррентной нейронной сети LSTM. Последующей задачей является проведение экспериментов по обучению нейросети (перебор гиперпараметров модели для достижения более успешного результата валидации модели). В конечном итоге необходима реализация приложения, в которое будет внедрена модель, и реагирующее на жестовые команды, выполняя заданные функции.

3.2 Приложение сбора данных

Более подробный алгоритм работы приложения сбора данных представлен ниже:

Рисунок 12 - Блок-схема основной задачи приложения сбора данных

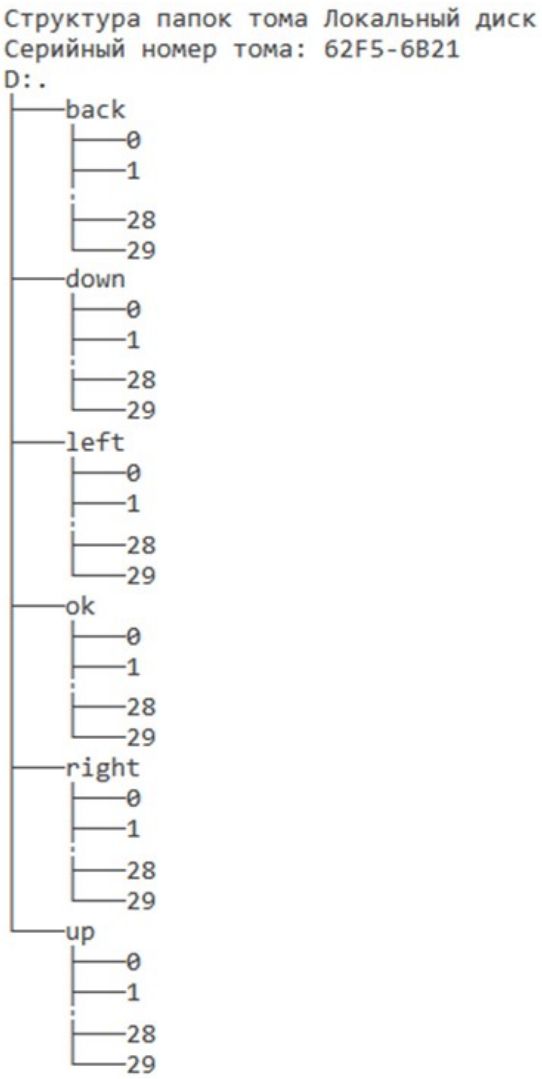


Рисунок 13 - Пример структуры папок

Следующий код выполняет данные операции:

```
def make_dirs(data_path, actions, no_sequences):
for action in actions:
  for sequence in range(no_sequences):
    try:
      os.makedirs(os.path.join(data_path, action, str(sequence)))
    except:
      pass
```

После создания директорий для хранения данных можно приступать к сбору данных.

Извлечение ключевых точек из изображения выполняет следующая функция process объекта класса Hands фреймворка MediaPipe,

Также, для «холостой» работы алгоритма, разработана функция, работающая аналогично, за исключением сохранения результата. Эту функцию можно назвать демонстрирующей.

Реализованный алгоритм сбора данных является самодостаточным и вышеописанные функции можно смело назвать «чистыми функциями», которые получают на вход изображение и создают файлы результатов на выходе. Таким образом, разработанные функции можно использовать в приложении, доступному обычному пользователю без опасений влияния приложения на операционную систему.

Таким образом, для «оборачивания» функций в пользовательское приложение был разработан с помощью инструмента QtDesigner следующий интерфейс (Рисунок - 14):

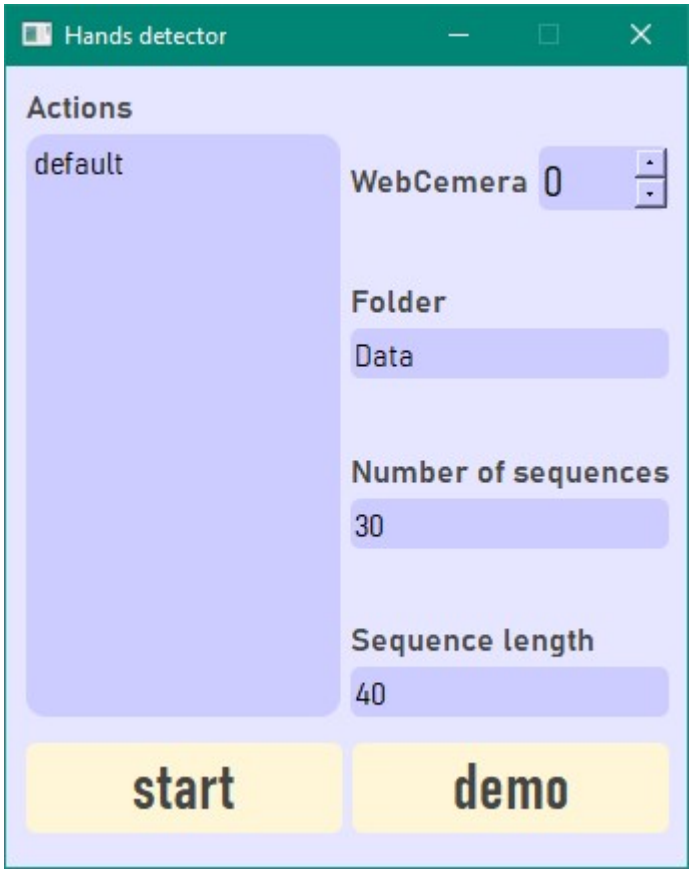


Рисунок 14 - Интерфейс приложения

С помощью библиотеки PyQt5 полученный с помощью QtDesigner интерфейс связан с функциями сбора данных. Разработанное приложение

работает следующим образом: Конфигурируются параметры сбора данных. После нажатия кнопки “Start” разворачивается окно, начинающее сбор данных, которое выглядит следующим образом:

Рисунок 15 - Окно сбора данных приложения

Приложение также может работать в демонстрационной режиме, не собирающем данные. Данный режим доступен по нажатию кнопки “Demo”.

3.3 Формулирование команд, сбор и обработка данных

Для выполнения задачи формулирования набора жестовых команд управления, необходимо определить интуитивно понятные человеку движения и достаточно простые для распознавания алгоритмом, во избежание нечётких срабатываний.

Так же жесты должно в полной мере удовлетворять базовые потребности в управлении. За пример был взят обычный пульт с стандартными кнопками: Вверх, Вниз, Влево, Вправо, Назад, ОК. Таким образом, были определены следующие движения рукой с ладонью направленной в сторону камеры (Рисунок - 16):

Рисунок 16 - Жесты управления

Во избежание непреднамеренных активаций команд было принято использовать так называемый «синхрожест», который представляет собой наполовину сжатый кулак (Рисунок - 17) и сопровождает управляющий жест от начала до конца.

Рисунок 17 - Пример синхрожеста

Собранные данные представляют собой набор из 6 жестов, по 30 записей, по 40 кадров, в сумме 7200 .npy файлов (Библиотеки Numpy для Python), который состоит из 126 чисел с плавающей точкой, описывающих 3D-координаты двух рук по 21 ключевой точке и выглядящие следующим образом:

```
[array([ 0.42938599,  0.56767941,  0.        ,  0.46548966,  0.5553869 ,
       -0.00987106,  0.49247044,  0.53347236, -0.01593102,  0.51416409,

      …
       0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
       0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
       0.        ]),
…
array([ 0.42413726,  0.5693543 ,  0.        ,  0.44785389,  0.53808248,
       -0.00742947,  0.46761057,  0.49912149, -0.01242292,  0.48233014,
      …
              0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
              0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
              0.        ])]
```

Обрабатывать и визуализировать было принято только одну правую руку с целью экономии вычислительных ресурсов, т.к. наличие двух рук увеличивает количество фич (от англ. feature - признак) вдвое и обучение

нейронной сети на данных большей размерности увеличивается на порядок, что уж говорить о большем количестве рук. Для демонстрации и решения поставленной задачи достаточно лишь одной руки (При наличии более высоких мощностей возможно легко адаптировать представленный алгоритм для обработки и визуализации двух и более рук).

Коллекция обнаруженных/отслеженных рук, где каждая рука представлена в виде списка из 21 ориентира руки, и каждый ориентир состоит из **x, y, z**. Где **x** и **y** нормализуются от 0.0 до 1.0 по ширине и высоте изображения соответственно. А **z** представляет глубину ориентира, причем глубина на запястье является началом координат, и чем меньше значение, тем ближе ориентир к камере. Величина **z** вычисляется примерно того же масштаба, **x** что и **y**. Ключевые точки в файлах расположены следующим образом (Рисунок - 18):

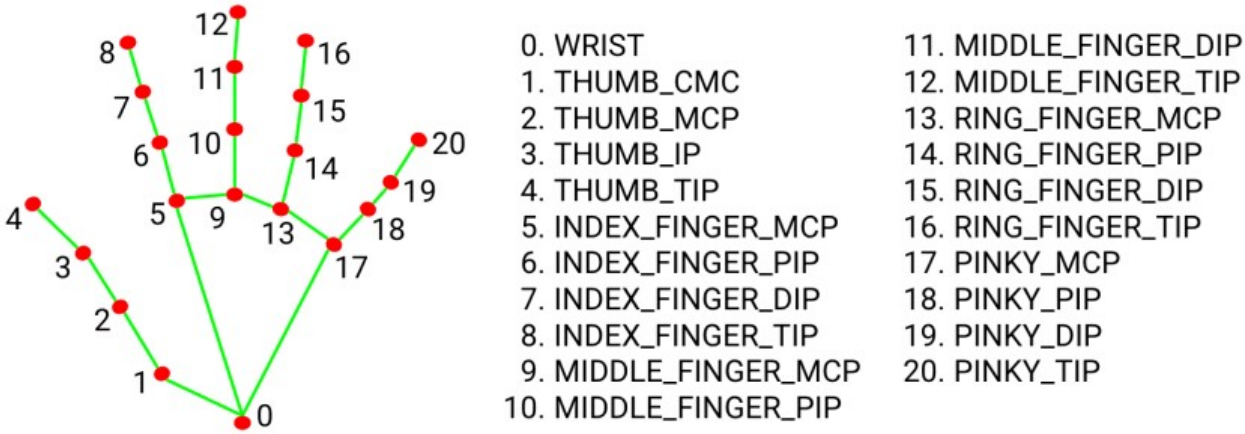


Рисунок 18 - Карта ключевых точек кисти руки

Следующей задачей является получение данных из файлов и проведение обработки.

Рисунок 19 - Блок-схема получения датасета

Разработку алгоритма необходимо начать с получения данных. Для этого необходимо указать относительный путь к папке где расположены все файлы.

Так как алгоритм заранее не знает какое количество жестов, сколько записей и их размер, необходимо явно это указать.Так же необходимо указать связи между ключевыми точками кисти руки.

Далее, с учётом известных параметров, данные извлекаются из файлов и помещаются в объект (переменную).

Данные операции выполняет следующий код:

```
import os
DATA_PATH = os.path.join('MP_Data\\Hands_Data')
actions = np.array([ 'up', 'down', 'left', 'right', 'ok', 'back'])
no_sequences = 30
sequence_length = 40
label_map = {label:num for num, label in enumerate(actions)}
parts = [
   [0, 1, 0, 5, 0, 17, 'green'],
   [5,9,9,13,13,17,'red'],
   [1,2,2,3,3,4,'blue'],
```

Перед тем как визуализировать данные необходимо их обработать. Произведя визуализацию можно увидеть отсутствие данных в некоторых кадрах, т.н. пробелы. Таким образом, для заполнения пробелов необходимо использовать интерполяцию, в частности линейную. Алгоритм интерполяции данных следующий:

Рисунок 20 - Блок-схема интерполяции данных

Данные, извлеченные из файлов, передаются в функцию. Далее происходит разделение данных по координатам на наборы X, Y и Z соответственно. В последствии каждый набор линейно интерполируется по всей последовательности. Эти операции выполняет следующая функция:

```
def interp_coords(x):
    coords = []
    for num,i in enumerate(x):
        if np.count_nonzero(i) != 0:
            coords.append([i,num])
    result = []
    for i in range(63):
        result.append(np.interp(range(40),[e[1] for e in coords],[e[0][i] for e in coords]))
    return np.array(result).transpose()
```

Рисунок 21 - Блок-схема получения визуализации

Затем можно приступать к визуализации. Для визуализации реализована функция, принимающая следующие параметры: координаты ключевых точек кисти руки, связи ключевых точек, флаги сохранения, фиксированного масштаба координатных осей и динамического изменения точки зрения. В функции для каждого кадра извлекаются ключевые точки и помещаются в трёхмерный график. Затем на график помещаются связи между ключевыми точками. После этого выбирается

масштаб осей автоматический или фиксированный, демонстрирующий всю область видимости ключевых точек, в зависимости от установленного флага fixed_axes. И выбирается динамическая или статическая точка зрения на график флагом dynamic_view. В зависимости от значения флага save результат сохраняется. В конечном итоге с помощью библиотеки MatPlotLib получаем визуализацию каждого отдельного кадра.

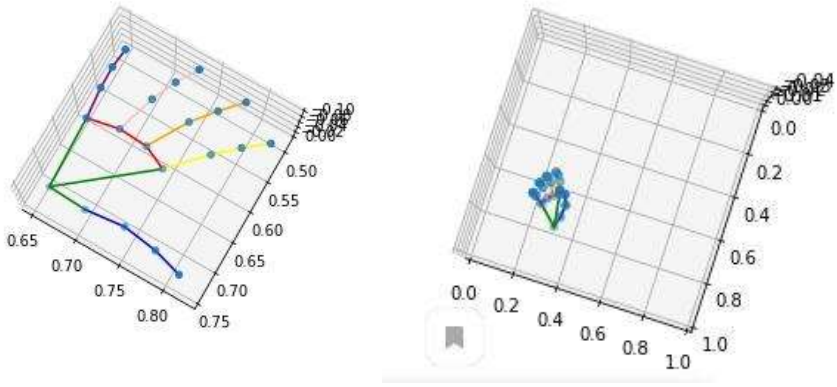


Рисунок 22 - Пример визуализации кадра

Код данной функции и используемых внутри неё:

```
    def draw_parts(ax, dx, dy, dz, parts):
        ax.plot3D([dx[parts[0]],dx[parts[1]]],                    [dy[parts[0]],dy[parts[1]]],
[dz[parts[0]],dz[parts[1]]], parts[6])
        ax.plot3D([dx[parts[2]],dx[parts[3]]],                    [dy[parts[2]],dy[parts[3]]],
[dz[parts[2]],dz[parts[3]]], parts[6])
        ax.plot3D([dx[parts[4]],dx[parts[5]]],                    [dy[parts[4]],dy[parts[5]]],
[dz[parts[4]],dz[parts[5]]], parts[6])

    def get_coords(X):
        dx = X[:63:3]
        dy = X[1:63:3]
        dz = X[2:63:3]
        return dx,dy,dz

    def draw_hand(x, parts, save=False, fixed_axes=False, dinamic_view=False):
        for num,i in enumerate(x):
            dx,dy,dz = get_coords(i)

            fig = plt.figure()
            ax = fig.add_subplot(111, projection='3d', label=num)
            ax.scatter(dx, dy, dz)

            for i in parts:
                draw_parts(ax, dx, dy, dz, i)
```

3.4 Обучение LSTM сети

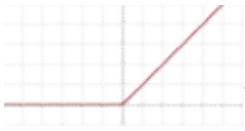
3.4.1 Конфигурация

Исходя из объёма и формата имеющихся данных (6 жестов, по 30 записей, по 40 кадров каждая) была сконфигурирована следующая модель рекуррентной нейронной сети LSTM:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_3 (LSTM)               (None, 40, 64)            48896

 lstm_4 (LSTM)               (None, 40, 128)           98816

 lstm_5 (LSTM)               (None, 64)                49408

 dense_3 (Dense)             (None, 64)                4160

 dense_4 (Dense)             (None, 32)                2080

 dense_5 (Dense)             (None, 6)                 198

=================================================================
Total params: 203,558
Trainable params: 203,558
Non-trainable params: 0
```

Рисунок - 23 Блок-схема получения визуализации

На протяжении всей структуры модели основной функцией активации является так называется функция ReLU (от англ. Rectified linear unit - Линейный выпрямитель или полулинейный элемент):



$$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geqslant 0 \end{cases}$$

Рисунок 24 - Функция активации ReLU

Эта функция является универсальной функцией обучения различного рода и форм нейронных сетей благодаря своей линейности. С помощью только этой функции активации можно добиться хорошего результата обучения независимо от количества слоёв нейронной сети.

Другой функцией активации является функция активации Softmax, выполняющая активацию в суммирующем финальном слое, выводя на выходе вероятности отношения классифицируемого объекта к i-тому классу. Softmax — это обобщение логистической функции для многомерного случая. Функция преобразует вектор z размерности K в вектор σ той же размерности, где каждая координата σ_i полученного вектора представлена вещественным числом в интервале [0,1] и сумма координат равна 1. Координаты σ_i вычисляются следующим образом:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

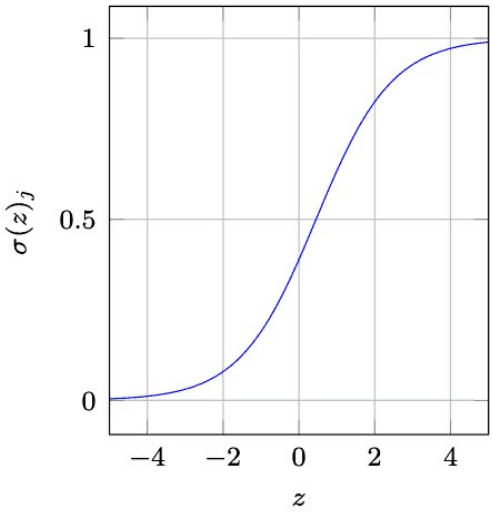


Рисунок 25 - Функция активации Softmax

В качестве входного слоя модели выступают элементы LSTM в количестве 40 на 64. На выходе модели находится суммирующий слой с функцией softmax, и имеющий 6 выходов, выводящих вероятности отношения классифицируемого объекта к i-тому классу от 0 до 1.

Весь код, выполняющий конфигурацию модели рекуррентной нейронной сети LSTM:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import TensorBoard

model = Sequential()
model.add(LSTM(64,         return_sequences=True,         activation='relu',
input_shape=(40,126)))
model.add(LSTM(128, return_sequences=True, activation='relu'))
```

```
        model.add(LSTM(64, return_sequences=False, activation='relu'))
        model.add(Dense(64, activation='relu'))
        model.add(Dense(32, activation='relu'))
        model.add(Dense(actions.shape[0], activation='softmax'))

        model.compile(optimizer='Adam',            loss='categorical_crossentropy',
metrics=['categorical_accuracy'])
```

3.4.2 Проведение экспериментов по обучению и валидации модели

Исходя из целей проекта метрикой оценивания модели будет являться Accuracy (от англ. Точность). Accuracy - это то, насколько близок или далек данный набор измерений (наблюдений или показаний) от их истинного значения.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Рисунок 26 - Формула вычисления Accuracy

Где:

TP = True positive; (положительный исход)

FP = False positive; (Ложноположительный исход)

TN = True negative; (Отрицательный исход)

FN = False negative (Ложноотрицательный исход)

Более наглядно эту формулу можно понять следующим образом. Учитывая, что возможны два варианта ответа системы и два правильных ответа, то всего возможно 4 исхода:

Реальные показания	Система отвечает «да»	Система отвечает «нет»
Да	Положительный исход	Ложноотрицательный исход
Нет	Ложноположительный исход	Отрицательный исход

Пример обучения нейронной сети:

```
Epoch 1/100
```

```
                    sentence.append(actions[np.argmax(res)])
                else:
                    sentence.append(actions[np.argmax(res)])

            if len(sentence) > 5:
                sentence = sentence[-5:]

            image = prob_viz(res, actions, image, colors)

        cv2.rectangle(image, (0,0), (640, 40), (245, 117, 16), -1)
        cv2.putText(image, ' '.join(sentence), (3,30),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

        cv2.imshow('Detector', image)

        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

Реализованное приложение имеет следующий интерфейс:

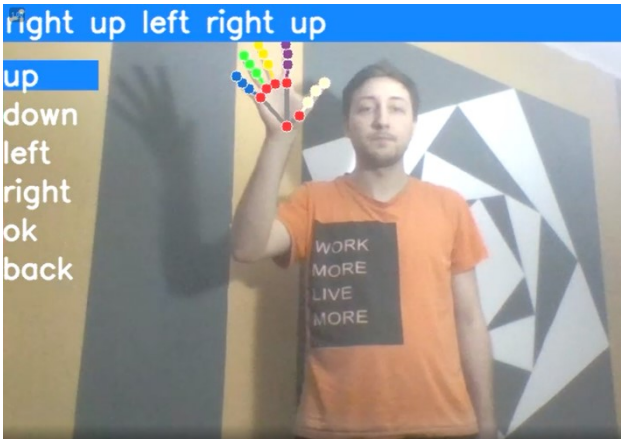


Рисунок 27 - Пример выполнения жестовой команды «Вверх»

Рисунок 28 - Пример выполнения жестовой команды «Вниз»

Рисунок 29 - Пример выполнения жестовой команды «Влево»

Рисунок 30 - Пример выполнения жестовой команды «Вправо»

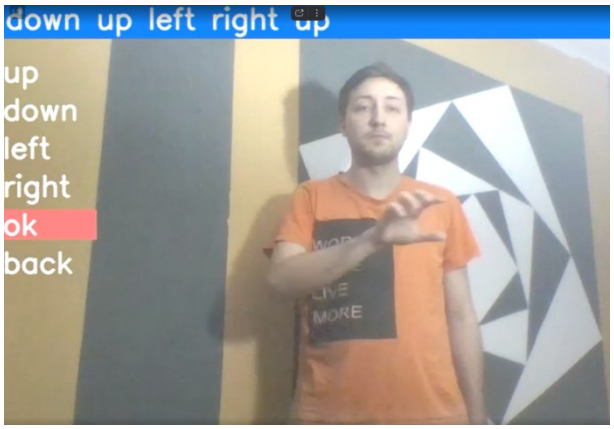


Рисунок 31 - Пример выполнения жестовой команды «ОК»

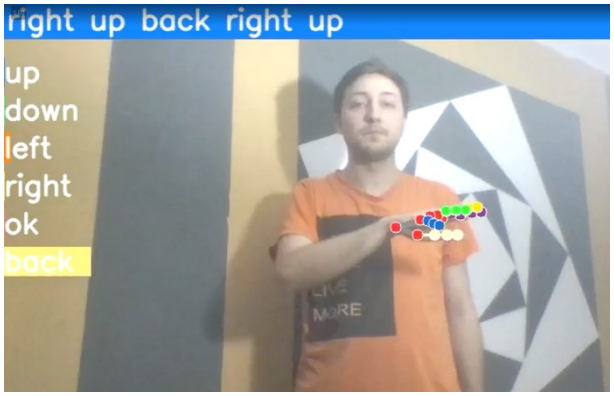


Рисунок 32 - Пример выполнения жестовой команды «Назад»

Как можно было видеть выше, реализованное приложение достаточно точно определяет команду. Единственным минусом на данный момент являются, выявленные на момент тестирования приложения, требования к вычислительным способностям компьютера и чёткости и скорости камеры. В следствии чего, в работе приложения возникают так называемые баги, вызванные уменьшением операционной системой выделенных на работу приложения ресурсов, замедляя работу приложения когда приложение работает в основном для него фоновом режиме. Приложение было разработано и протестировано на ноутбуке ASUS Tuf-Gaming FX505DY с процессором AMD Ryzen 5 3550H 4 ядра 8 потоков и базовой частотой 2.1 ГГц, 8 гигабайтами оперативной памяти DDR4 SODIMM. Увеличение вычислительной мощности компьютера и повышения приоритета процесса приложения значительно повышает его производительность, что было выявлено при тестировании приложения на ноутбуке HP с процессором Intel Core i7 10700 8 ядер 16 потоков и базовой частотой 3.8 ГГц и оперативной памятью 32 гигабайта DDR4 SODIMM.

запястье лежит на столе. Этот метод не разрешен. Дисплей должен находиться прямо перед пользователем..

4.5 Требования к организации пространства

Экраны мониторов — не единственный источник вредного электромагнитного излучения. Разработчикам мониторов уже давно удалось их преодолеть. Меньше внимания уделяется вредным паразитным выбросам от боковых и задних стенок оборудования. В современных компьютерных системах эти области являются наиболее опасными.

Монитор компьютера должен располагаться так, чтобы задняя стенка была обращена не к человеку, а к стене комнаты. В машинном зале с несколькими компьютерами рабочее место должно располагаться на периферии помещения, оставляя центр свободным. При этом также необходимо проверить, чтобы на каждом рабочем месте не было прямых отражений от внешних источников света. Зачастую добиться этого для всех работ одновременно довольно сложно. Возможное решение — использование штор на окнах и грамотное размещение общего и местного освещения от искусственных источников света..

Сильными источниками электромагнитных излучений являются устройства бесперебойного питания. Располагать их следует как можно дальше от посадочных мест пользователей.

5 ЭНЕРГОСБЕРЕЖЕНИЕ

Энергосбережение — реализация правовых, организационных, научных, производственных, технических и экономических мер, направленных на эффективное (рациональное) использование (и экономное расходование) топливно-энергетических ресурсов и на вовлечение в хозяйственный оборот возобновляемых источников энергии. При разработке ПО самые большие траты приходятся на электроэнергию, потребляемую компьютерами.

5.1 Основные способы экономии энергии

Ключевыми мероприятиями по оптимизации энергопотребления в области освещения являются:

- Максимальное возможное использование солнечно света (повышение прозрачности и увеличение площади окон, дополнительные окна);
- повышение способности к отражению (белые стены и потолок);
- оптимальное размещение источников света (местное освещение, направленное освещение);
- использование осветительных устройств в случаях острой необходимости;
- увеличение показателей светоотдачи имеющихся источников (замена люстр, плафонов, удаление грязи с плафонов, применение более эффективных отражателей);
- замена ламп накаливания на энергосберегающие (люминесцентные, компактные люминесцентные, светодиодные);
- применение устройств управления освещением (датчики движения и акустические датчики, датчики освещенности, таймеры, системы дистанционного управления);

- внедрение автоматизированной системы диспетчерского управления наружным освещением (АСДУ НО);
- установка интеллектуальных распределённых систем управления освещением (минимизирующих затраты на электроэнергию для данного объекта).

5.2 Энергосберегающие режимы работы компьютера

Режим энергосбережения - это режим энергосбережения, который может быстро возобновить работу в обычном режиме энергосбережения (обычно в течение нескольких секунд) по запросу пользователя.

Перевод компьютера в спящий режим напоминает нажатие кнопки "Пауза" на проигрывателе DVD — компьютер сразу стопорит все операции и полон решимости к возврату в рабочий режим при нужды.

Режим гибернации - это режим с низким энергопотреблением, предназначенный в основном для ноутбуков. Когда вы переходите в спящий режим, все документы и открытые функции сохраняются в памяти, и компьютер переходит в спящий режим, а когда вы переходите в спящий режим, все открытые документы и программы сохраняются на жестком диске, а затем компьютер выключается. Из всех режимов энергосбережения, используемых в операционной системе Windows, для поддержания спящего режима требуется наименьшее количество энергии. Если вы не хотите использовать ноутбук в течение длительного времени и не можете зарядить аккумулятор, рекомендуется перевести ноутбук в спящий режим.

Гибридный спящий режим — это режим, который разработан преимущественно для настольных компьютеров. Гибридный спящий режим сочетает в себе спящий режим и режим гибернации, поскольку все открытые документы и программы сохраняются в памяти и на жестком диске и компьютер переводится в режим пониженного потребления

электроэнергии. При неожиданном сбое питания операционная система Windows может легко восстановить данные с диска. Если гибридный спящий режим включен, переход в спящий режим автоматически переводит компьютер в гибридный спящий режим. На настольных компьютерах гибридный спящий режим обычно активен по умолчанию.

Таким образом, наиболее эффективная экономия энергии достигается, когда режим гибернации используется в сочетании с спящим режимом или в гибридном спящем режиме.

5.3 Энергоэффективность в серверных помещениях

Техническое обслуживание включает в себя серверное оборудование и системы кондиционирования воздуха. Необходимо поддерживать правильные настройки. Например, не следует устанавливать слишком низкую температуру системы кондиционирования, которая не нужна для работы сервера. Необходимо обратить внимание на расположение перемычек в серверном шкафу. Обычно со временем клубок кабелей копится, потоки воздуха начинают испытывать излишнее сопротивление. Пустые юниты в стойках стоит закрывать панелями-заглушками, чтобы холодный воздух не перемешивался с горячим. Нужно вовремя чистить и мыть оборудование, т.к. грязный наружный блок кондиционера или грязные воздушные фильтры во внутренних блоках существенно снижают теплообмен. Если не следить за всем выше перечисленным, существенно увеличиваются энергозатраты (компрессор и вентиляторы вынуждены работать более производительно).

5.4 Стандарты технологий энергосбережений

В начале 1990-х Агентство по охране окружающей среды (EPA) начало кампанию по сертификации энергоэффективных персональных компьютеров и периферийных устройств. Ваш компьютер или монитор

должен снизить энергопотребление до 30 Вт или более в течение продолжительных периодов бездействия. Система, отвечающая этим требованиям, может получить сертификат Energy Star.

5.4.1 Стандарт (Advanced Power Management — APM) расширенной системы управления питанием

Разработан компанией Фирмы intel общо с Microsoft и определяет ряд интерфейсов меж аппаратными средствами управления питанием и операционной системой компьютера. На сто процентов реализованный эталон APM дает возможность автоматом переключать компьютер между пятью состояниями в большой зависимости от текущего состояния системы. Каждое последующее состояние в приведённом ниже списке характеризуется убавлением употребления энергии.

- Full On

Система активирована на полную мощность.

- APM Suspend (режим приостановки).

Система не работает, большинство устройств пассивны. Тактовый генератор центрального процессора остановлен, а параметры функционирования хранятся на диске и при необходимости могут быть считаны в память для восстановления работы системы. Чтобы запустить систему из этого состояния, требуется некоторое время.

- APM Standby (резервный режим)

Система не работает, Большинство устройств находятся в состоянии употребления малой мощности. Работа тактового генератора центрального микропроцессора возможно замедлена либо остановлена, но нужные характеристики функционирования хранятся в памяти. Юзер либо операционная система могут запустить компьютер из этого состояния практически моментально.

- APM Enabled

Система работает, некоторые устройства являются объектами управления для системы управления кормлением. Неиспользуемые устройства могут являться выключены, возможно также остановлена либо замедлена (то есть снижена тактовая частота) работа тактового генератора центрального микропроцессора.

- Off (система отключена).

Система не работает. Источник питания отключён.

Для реализации режима APM требуется как аппаратное, так и программное обеспечение. Блок питания ATX можно управлять с помощью сигнала Power_On и дополнительного шестиконтактного разъема питания. (Необходимые команды выдаются программным обеспечением). Производители также интегрируют это управление с другими системными компонентами, такими как материнские платы, мониторы и жесткие диски. Операционные системы (например, Windows), поддерживающие APM, запускают программы управления питанием при соответствующих событиях и следят за действиями пользовательских и прикладных программ. Однако операционная система не отправляет сигналы управления питанием непосредственно на оборудование. Система может иметь множество различных аппаратных и программных функций, используемых при запуске функций APM. Чтобы решить проблему интерфейса этих инструментов, в операционной системе и аппаратном обеспечении предусмотрен специальный абстрактный уровень, который облегчает связь между различными элементами архитектуры.

При запуске операционной системы загружается программа-драйвер APM, которая связана с различными прикладными программами и функциями программного обеспечения. Именно они запускают действия по управлению питанием, при этом все APM-совместимое оборудование

подключается к системному BIOS. Драйвер APM и BIOS напрямую связаны друг с другом; именно это соединение операционная система использует для управления аппаратными режимами.

Для работы инструментов APM требуется стандарт, который поддерживается схемами, встроенными в определенные системные аппаратные устройства, системный BIOS и операционную систему с драйвером APM. Если хотя бы один из этих компонентов отсутствует, APM не будет работать.

5.4.2 Стандарт (Advanced Power Management—APM) расширенной системы управления питанием

Расширенный интерфейс конфигурации и питания (ACPI) был впервые представлен в BIOS и современных операционных системах Windows 98 и более поздних версий. Если BIOS компьютера поддерживает систему ACPI, все управление питанием переносится на операционную систему. Это упрощает настройку параметров, которые все находятся в одном месте - в операционной системе.

5.4.3 Стандарт DPMS (Display Power Management Signaling - система управления питанием сигналов монитора)

Стандарт ассоциации VESA. Определяет состав сигналов, которые компьютер отправляет на монитор, когда система переходит в энергосберегающий режим из состояния ожидания. В этих процедур системы, драйвер берет на себя контроль, который посылает сигналы, совпадающие с помощью видеокарты. Когда вы нажимаете клавишу на клавиатуре или переместите мышь, монитор переключается на нормальный режим работы.

6 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

В этом разделе содержится информация, из которой выводится целесообразность (или непрактичность) создания программного обеспечения. Содержательный анализ затрат и результатов проекта. Позволяет определить, стоит ли инвестировать в предлагаемый проект.

6.1 Оптимальные характеристики оборудования

Как инструмент разработки программ использовался ноутбук ASUS Tuf-Gaming FX505DY со следующими ключевыми техническими характеристиками:

- Процессор AMD Ryzen 5 CPU 3550H @ 2.1GHz
- Встроенный дисплей с разрешением 1920x1080
- ОЗУ DDR4 SODIMM 8 Гбайт

Отметим пункты для значительного увеличения скорости разработки программ:

- Наличие дополнительного монитора
- Мониторы с высоким разрешением экрана
- Высокопроизводительный многоядерный процессор
- Большой размер ОЗУ (16Гбайт - 32Гбайт)
- Наличие SSD в качестве основного дискового пространства (для операционной системы и запуска программ)

Указанные пункты помогут уменьшить время разработки программ за счет увеличения скорости работы сборщика проектов, возможности одновременного доступа к нескольким источникам информации без необходимости переключения внимания и скорости проведения тестов разработанного ПО за счет возможности параллельной работы сразу с несколькими эмуляторами вместо последовательной в случае работы с устаревшим оборудованием.

Минимальные системные требования:

- Процессор 2ГГц
- Разрешение экрана 1024x768
- ОЗУ 4Гбайта
- Наличие монитора, клавиатуры и мыши

6.2 Расчет себестоимости программного обеспечения

При расчёте себестоимости необходимо учитывать ряд технико-экономических показателей, к которым относятся сроки разработки, размер и объем программного кода.

Общая стоимость создания скрипта складывается из нескольких составляющих. При расчёте все эти составляющие должны быть приведены к единице измерения для обеспечения единообразия и удобства сравнения. В качестве такой единицы удобнее рассматривать денежные единицы или человеко-часы.

При разработке программных комплексов наибольшее значение в составе затрат имеют следующие составляющие:

1. затраты на подготовку и применение технологии и программных средств автоматизации разработки программ;
2. затраты на проектирование, программирование, отладку и испытания программ согласно требований пользователя или заказчика;
3. затраты на ЭВМ, используемые при разработке данного программного продукта;
4. затраты на изготовление опытного образца программного продукта как продукции производственно–технического назначения;
5. затраты на подготовку и повышение квалификации специалистов–разработчиков.

Затраты типа 1 и 4 можно исключить, учитывая условия и характер разработки данного программного продукта. Затраты типа 5 тоже не учитываются, так как их наиболее трудно учитывать и формализовать в конкретной работе. Имеются и дополнительные составляющие стоимости, которые не учитываются исходя из условий работы. К этим составляющим относятся: заработная плата, мебель, коммунальные услуги, охрана, противопожарные мероприятия, налоги и т.д.

Для определения составляющих затрат труда воспользуемся расчетом условного числа выполняемых строк программного кода в программе:

$$Q = q \cdot c \cdot (1 + p), \quad (1)$$

где c = 1,2 — коэффициент увеличения затрат, характеризует увеличение затрат труда вследствие недостаточно полного описания задачи, уточнений и некоторой доработки. Этот коэффициент может принимать значения от 1,2 до 5;

p = 0.08 — коэффициент коррекции программы в ходе ее разработки, принимает значения от 0.05 до 0.1;

q = 1604— число строк программного кода.

В итоге получаем, что условное число операторов в программе принимает значение:

$$Q = 1604 \cdot 1{,}2 \cdot (1 + 0{,}08) = 2079$$

Важной составляющей при расчете себестоимости программного продукта является коэффициент квалификации k. Коэффициент квалификации берется из таблицы коэффициентов, приведенной ниже.

Таблица 1 - Коэффициенты квалификации

Опыт работы	Коэффициент
До 2–х лет	0,8
2 — 3 года	1,0
3 — 5 лет	1,1–1,2

Продолжение таблицы 1

Опыт работы	Коэффициент
5 — 7 лет	1,3–1,4
Больше 7 лет	1,5–1,6

Согласно данным таблицы, необходимый коэффициент квалификации принимаем равным k = 1,2.

Для определения трудоемкости разработки приложения необходимо рассчитать следующие виды затрат:

- затраты труда на подготовку описания задачи;
- затраты труда на разработку блок–схемы приложения;
- затраты труда на программирование по блок–схеме;
- затраты труда на отладку кода;
- затраты труда на подготовку документации/описания.

Затраты труда на подготовку описания задачи рассчитываются по следующей формуле:

$$t_{\text{оп}} = \frac{T_{\min} + 4 \cdot T_{\text{наиб.вероят.}} + T_{\max}}{6}, \quad (2)$$

где $T_{\min}$ — минимально возможная трудоемкость выполнения работы;

$T_{\text{наиб.вероя т.}}$ — наиболее вероятная трудоемкость;

$T_{\max}$ — максимально возможная трудоемкость.

По формуле 2 получаем, что затраты составляют

$$t_{\text{оп}} = \frac{8 + 4 \cdot 14 + 20}{6} = 14 \quad \text{(чел/час)}.$$

Затраты труда на разработку блок–схемы алгоритма:

$$t_{\text{ал}} = \frac{Q \cdot B}{(75 \div 85) \cdot k}, \quad (3)$$

где B — коэффициент увеличения затрат в зависимости от качества постановки задачи [1,2...5];

k — коэффициент квалификации берется из диапазона [75…85], (примем его равным 80):

$$t_{\text{ал}} = \frac{Q \cdot B}{(75 \div 85) \cdot k} = \frac{1604 \cdot 1{,}2}{80 \cdot 1.2} \approx 20 \quad \text{(чел/час)}.$$

Затраты труда на программирование по блок–схеме рассчитывается по формуле (коэффициент квалификации берется из диапазона [60…75], примем его равным 70):

$$t_{\text{пр}} = \frac{Q}{(60 \div 75) \cdot k} \qquad (4)$$

$$t_{\text{пр}} = \frac{Q}{(60 \div 75) \cdot 1{,}2} = \frac{1604}{24} \approx 67\,(\text{чел/час}).$$

Затраты труда на отладку кода (коэффициент квалификации берется из диапазона [40…50], примем его равным 40):

$$t_{\text{отл}} = \frac{Q}{(40 \div 50) \cdot k} \qquad (5)$$

$$t_{\text{отл}} = \frac{Q}{(40 \div 50) \cdot k} = \frac{1604}{40 \cdot 1{,}2} \approx 33\,(\text{чел/час}).$$

Затраты труда на подготовку документации:

$$t_{\text{Д}} = t_{\text{пр}} + t_{\text{оф}}\,(\text{чел/час}), \qquad (6)$$

где $t_{\text{пр}}$ — затраты труда на подготовку

$t_{\text{оф}}$ — время на оформление документов

$$t_{\text{пр}} = \frac{Q}{(150 \div 200) \cdot k} = \frac{1604}{175 \cdot 1{,}2} = 8\,,$$

$$t_{\text{оф}} = t_{\text{пр}} \cdot 0{,}75\,(\text{чел/час}),$$

$$t_{\text{оф}} = 6$$

В итоге получаем:

$$t_{\text{Д}} = 14\,(\text{чел/час}).$$

Тогда итоговая трудоемкость разработки данного программного обеспечения равна:

$$t_{\Sigma} = t_{\text{оп}} + t_{\text{ал}} + t_{\text{пр}} + t_{\text{отл}} + t_{\text{Д}} \text{ (чел/час)},$$

$$t_{\Sigma} = 14 + 20 + 67 + 33 + 14 = 148 \text{ (чел/час)}.$$

Это время приблизительно равняется одному месяцу при длине рабочего дня в 8 часов, что соответствует реальному времени разработки данного программного продукта.

При расчете затрат на разработку программного обеспечения необходимо разделить на составляющие: заработная плата (основная и дополнительная), отчисления на социальные нужды, эксплуатационные затраты (электроэнергия, техническое обслуживание и текущий ремонт), накладные расходы, материалы и комплектующие.

Основная заработная плата рассчитывается по формуле:

$$З_{\text{осн}} = \frac{t_{\Sigma}}{t_{\text{ср}} \cdot 8} \cdot TC \text{ (руб.)}, \quad (7)$$

где t_{Σ} – суммарные затраты труда, 148(чел/час);

$t_{\text{ср}}$ – среднее число рабочих дней в месяце: (365–107)/12 = 21,5 дней;

Тарифная ставка (ТС) представляет собой минимальный размер оплаты труда (МРОТ), увеличенный в зависимости от тарифного коэффициента $k_{\text{т}}$, соответствующего данному разряду работ. Для 13–ого разряда работ, который соответствует работе программиста, тарифный коэффициент равен 3,04.

$$\text{ТС} = \text{МРОТ} \cdot k_{\text{т}} = 480 \cdot 3{,}04 = 1459{,}2 \text{ (руб/мес)},$$

$$З_{\text{осн}} = \frac{148}{21{,}5 \cdot 8} \cdot 1459{,}2 = 1255{,}59 \text{ (руб.)},$$

$$t_{\text{з}} = \frac{t_{\Sigma}}{t_{\text{ср}} \cdot 8} = \frac{1604}{21{,}5 \cdot 8} = 0{,}84 \text{ (мес.)},$$

где $t_{\text{з}}$ - время, затраченное на создание программного обеспечения (в месяцах).

Надбавка за стаж составляет 5% от основной заработной платы

$$З_{доп} = 5\% \cdot З_{осн} = 0{,}05 \cdot 1255{,}59 = 62{,}7795 \text{ (руб.)}$$

Премия 30 % от основной заработной платы за успешно выполненное задание в срок

$$З_{прем} = 30\% \cdot З_{осн} = 0{,}3 \cdot 1255{,}59 = 376{,}677 \text{ (руб.)}.$$

Отчисление на социальное страхование составляют 35% от всей заработной платы:

$$З_{соц.страх.} = 35\% \cdot (З_{осн} + З_{доп} + З_{прем}) = 0{,}35 \cdot (1255{,}59 + 62{,}7795 + 376{,}677) = 593{,}27 \text{ (руб.)}$$

Эксплуатационные затраты возникают при эксплуатации ЭВМ:

Стоимость электроэнергии:

$$C_{ээ} = M \cdot к_{з} \cdot F_{эф} \cdot C_{квт.ч.}, \quad (8)$$

где M — мощность ЭВМ (0,06 кВт);

$к_{з}$ — коэффициент загрузки (0,7);

$C_{кВт.ч.}$ — стоимость 1 кВт час электроэнергии (0,165 руб.);

$F_{эф}$ — эффективный фонд, рассчитывается по формуле:

$$F_{эф} = Д_{ном} \cdot 8 \cdot \left(1 - \frac{f}{100}\right), \quad (9)$$

где $Д_{ном}$ — номинальное число рабочих дней в году (255);

f — планируемый процент времени на ремонт ЭВМ (2%);

тогда

$$F_{эф} = 255 \cdot 8 \cdot \left(1 - \frac{2}{100}\right) = 1999 \text{ (час.)},$$

тогда стоимость электроэнергии составит:

$$C_{ээ} = 0{,}06 \cdot 0{,}7 \cdot 1999 \cdot 0{,}165 = 13{,}85 \text{ (руб.)}.$$

Это стоимость электроэнергии за год, а за период разработки программного обеспечения стоимость электроэнергии составит:

$$C_{ээ.разраб.} = C_{ээ} \cdot \frac{t_{\Sigma}}{Д_{ном} \cdot 8}, \quad (10)$$

где t_{Σ} — суммарные затраты труда, 1273 (чел/час);

$Д_{ном}$ — номинальное число рабочих дней в году (255);

$C_{ээ}$ — стоимость электроэнергии за год.

Тогда стоимость электроэнергии за период разработки программного обеспечения составит:

$$C_{ээ.разраб.} = 13{,}85 \cdot \frac{148}{255 \cdot 8} = 1\,(\text{руб.}).$$

Предположительная стоимость компьютера с необходимыми параметрами будет составлять 2000 (руб.).

Техническое обслуживание и ремонт составляют 2.5% от стоимости компьютера, тогда получаем:

$$C_{то} = 2{,}5\% \cdot C_{ЭВМ} = 0{,}025 \cdot 2000 = 50\,(\text{руб.}).$$

Это стоимость и обслуживание технического ремонта за год, а за период разработки стоимость технического обслуживания и текущего ремонта составит:

$$C_{то.разраб.} = C_{то} \cdot \frac{t_{\Sigma}}{Д_{ном} \cdot 8} = 25 \cdot \frac{148}{255 \cdot 8} = 1{,}8\,(\text{руб.}),$$

где t_{Σ} — суммарные затраты труда, 1273 (чел.час);

$Д_{ном}$ — номинальное число рабочих дней в году (255).

Накладные расходы составляют 10% от основной заработной платы:

$$C_{накл} = З_{осн} \cdot 0{,}1 = 1255{,}59 \cdot 0{,}1 = 125{,}6\,(\text{руб.}).$$

Материалы и комплектующие составляют 1.5% от стоимости оборудования:

$$C_{МиК} = 1{,}5\% \cdot C_{ЭВМ} = 0{,}015 \cdot 2000 = 30\,(\text{руб.})$$

Таким образом, суммарные расходы на разработку программного обеспечения составили:

$$C_{\Sigma} = З_{осн} + З_{доп} + З_{прем} + З_{соц.страх.} + C_{ээ.разраб} + C_{накл} + C_{МиК} + C_{ТО.Разраб} \quad (\text{руб.}) \quad (11)$$

$$C_{\Sigma} = 1255{,}59 + 62{,}78 + 376{,}677 + 593{,}27 + 1 + 1{,}8 + 30 + 125{,}6 = 2446{,}717 \quad (\text{руб.})$$

Таблица 2 - Смета затрат а разработку

Составляющие себестоимости	Сумма, BYN
Зарплата	1255,59
Социальное страхование	593,27
Эксплуатационные расходы	1
Накладные расходы	125,6
Материалы и комплектующие	30
Техническое обслуживание	1,8
Итого	2007,26

Согласно данным таблицы 2 себестоимость разработанного программного продукта составляет 2007,26 (руб.).