Наименование ВУЗа

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе по дисциплине «»**

**по теме «Распознавание текста с изображения документа «Страховое свидетельство РФ»»**

Выполнил:

студент группы

Фамилия И.О.

Проверил:

преподаватель кафедры

Фамилия И.О.

# **СОДЕРЖАНИЕ**

[СОДЕРЖАНИЕ 2](#_Toc477106948)

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ 3](#_Toc477106949)

[ВВЕДЕНИЕ 4](#_Toc477106950)

[1. РАСЧЁТНАЯ ЧАСТЬ 5](#_Toc477106951)

[1.1. Выбор метода распознавания текста 5](#_Toc477106952)

[1.2. Предварительная обработка изображения 6](#_Toc477106953)

[1.3. Поиск сегментов связности 8](#_Toc477106954)

[1.4. Сопоставление сегмента с шаблонами символов 9](#_Toc477106955)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 10](#_Toc477106956)

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Разработать программу, считывающую текст с изображения документа и выводящую ее на экран.

Требования:

• Программа должна быть разработана на языке python.

• Должна поддерживаться работа из консоли. Графический интерфейс не нужен.

• Программа должна запускаться как на Windows, так и на Linux (Ubuntu).

• Код должен быть сопровожден комментариями (необходимо для сдачи курсовой).

• Должна быть пояснительная записка с описанием выбранных методов, алгоритмов и решений при разработке.

• Программа должна обладать следующими функциональными требованиями:

o Импорт изображения в формате jpeg или любом другом распространенном. Программа должна работать с изображениями документа «СНИЛС» – страховое свидетельство РФ.

o Выделения областей, содержащих текстовую информацию.

o Распознавание текста (поля номер СНИЛС, ФИО, Дата и место рождения, пол, дата регистрации). Желательно использование технологий нейронных сетей, но любые другие решения также возможны.

o Вывод распознанной информации на экран с перечислением полей.

o Погрешность распознавания – не менее 70%.

o Опционально – возможность сохранения данных в файл.

# ВВЕДЕНИЕ

Несмотря на то, что в настоящее время большинство документов составляется на компьютерах, задача создания полностью электронного документооборота ещё далека до полной реализации. Как правило, существующие системы охватывают деятельность отдельных организаций, а обмен данными между организациями осуществляется с помощью традиционных бумажных документов.

Задача перевода информации с бумажных на электронные носители актуальна не только в рамках потребностей, возникающих в системах документооборота. Современные информационные технологии позволяют нам существенно упростить доступ к информационным ресурсам, накопленным человечеством, при условии, что они будут переведены в электронный вид.

Наиболее простым и быстрым является сканирование документов с помощью сканеров. Результат работы является цифровое изображение документа – графический файл. Более предпочтительным, по сравнению с графическим, является текстовое представление информации. Этот вариант позволяет существенно сократить затраты на хранение и передачу информации, а также позволяет реализовать все возможные сценарии использования и анализа электронных документов. Поэтому наибольший интерес с практической точки зрения представляет именно перевод бумажных носителей в текстовый электронный документ.

**Оптическое распознавание символов** (англ. *optical character recognition, OCR*) — механический или электронный перевод изображений рукописного, машинописного или печатного текста в текстовые данные — последовательность кодов, использующихся для представления символов в компьютере (например, в текстовом редакторе).

**Задача оптического распознавания**

- Перевод документов, научных публикаций, социальной информации, исторических изданий в электронный вид.

- Классификация документов.

- Накопление и хранение электронных документов.

**История**

* 1929 году - Густав Таушек (*Gustav Tauschek*) получил патент на метод оптического распознавания текста в Германии;
* 1933 год - Гендель (*Paul W. Handel*) получил патент на свой метод в США ;
* 1935 год – Г. Таушек также получил патент США на свой метод;
* 1950 год - Дэвид Х. Шепард (*David H. Shepard*) - построил машину, решающую задачу преобразования печатных сообщений в машинный язык для обработки компьютером.
* 1955 год - Первая коммерческая система была установлена на «Ридерс Дайджест»
* 1965 год - «Ридерс Дайджест» и «Ар-Си-Эй» начали сотрудничество с целью создать машину для чтения документов, использующую оптическое распознавание текста, предназначенную для оцифровки серийных номеров купонов «Ридерс Дайджест», вернувшихся из рекламных объявлений.
* 1965 год - Почтовая служба Соединённых Штатов для сортировки почты использует машины, работающие по принципу оптического распознавания текста, созданные на основе технологий, разработанных исследователем Яковом Рабиновым.
* 1971 год - Почта Канады использует системы оптического распознавания символов
* 1974 год - Рэй Курцвейл создал компанию «Курцвейл Компьютер Продактс», и начал работать над развитием первой системы оптического распознавания символов, способной распознать текст, напечатанный любым шрифтом.
* 1978 год - Компания «Курцвейл Компьютер Продактс» начала продажи коммерческой версии компьютерной программы оптического распознавания символов.
* 1992 год – Начало продажи первой коммерчески успешной программой, распознающей кириллицу, «AutoR» российской компании «ОКРУС» (ОС DOS).
* Конец 60-х годов – разработка и испытание шрифтонезависимого алгоритма распознования текста выпускниками МФТИ, биофизиками: Г. М. Зенкиным и А. П. Петровым

При создании электронных библиотек и архивов путем перевода книг и документов в цифровой компьютерный формат, при переходе предприятий от бумажного к электронному документообороту, при необходимости отредактировать полученный по факсу документ используются системы оптического распознавания символов.

С помощью сканера несложно получить изображение страницы текста в графическом файле.

Однако для получения документа в формате текстового файла необходимо провести распознавание текста, т. е. преобразовать элементы графического изображения в последовательности текстовых символов.

-Сначала необходимо распознать структуру размещения текста на странице: выделить колонки, таблицы, изображения и т. д.

-Далее выделенные текстовые фрагменты графического изображения страницы необходимо преобразовать в текст.

**Структурный метод распознавания**

При распознавании документов с низким качеством печати (машинописный текст, факс и т. д.) используется метод распознавания символов по наличию в них определенных структурных элементов (отрезков, колец, дуг и др.).

Любой символ можно описать через набор параметров, определяющих взаимное расположение eгo элементов. Например, буква «Н» и буква «И» состоят из трех отрезков, два из которых расположены параллельно друг другу, а третий соединяет эти отрезки. Различие между буквами в величине улов, которые составляет третий отрезок с двумя другими. При распознавании структурным методом в искаженном символьном изображении выделяются характерные детали и сравниваются со структурными шаблонами символов. В результате выбирается тот символ, для которого совокупность всех структурных элементов и их расположение больше всего соответствуют распознаваемому символу.

**Программы распознавания текста**

Преобразованием графического изображения в текст занимаются специальные программы распознавания текста (Optical Character Recognition - OCR).

Современная OCR должна уметь многое: распознавать тексты, набранные не только определенными шрифтами, но и самыми экзотическими, вплоть до рукописных. Уметь корректно работать с текстами, содержащими слова на нескольких языках, корректно распознавать таблицы. И самое главное — корректно распознавать не только четко набранные тексты, но и такие, качество которых, мягко говоря, далеко от идеала. Например, текст с пожелтевшей газетной вырезки или третьей машинописной копии. Само собой, распознать текст — это еще полдела. Не менее важно обеспечить возможность сохранения результата в файле популярного текстового (или табличного) формата — скажем, формата Microsoft Word.

Наиболее распространенные системы оптического распознавания символов, например, **ABBYY FineReader**и **CuneiForm**от Cognitive, используют как растровый, так и структурный методы распознавания. Кроме того, эти системы являются «самообучающимися» (для каждого конкретного документа они создают соответствующий набор шаблонов символов) и поэтому скорость и качество распознавания многостраничного документа постепенно возрастают.

Существует системы Online распознавания текста: **Online OCR** и **FineReader Online** (http://www.onlineocr.ru, http://finereader.abbyyonline.com)

Теория распознава́ния о́браза — раздел информатики и смежных дисциплин, развивающий основы и методы классификации и идентификации предметов, явлений, процессов, сигналов, ситуаций и т. п. объектов, которые характеризуются конечным набором некоторых свойств и признаков. Такие задачи решаются довольно часто, например, при переходе или проезде улицы по сигналам светофора. Распознавание цвета загоревшейся лампы светофора и знание правил дорожного движения позволяет принять правильное решение о том, можно или нельзя переходить улицу.

Необходимость в таком распознавании возникает в самых разных областях — от военного дела и систем безопасности до оцифровки аналоговых сигналов.

Проблема распознавания образа приобрела выдающееся значение в условиях информационных перегрузок, когда человек не справляется с линейно-последовательным пониманием поступающих к нему сообщений, в результате чего его мозг переключается на режим одновременности восприятия и мышления, которому такое распознавание свойственно.

Неслучайно, таким образом, проблема распознавания образа оказалась в поле междисциплинарных исследований - в том числе в связи с работой по созданию искусственного интеллекта, а создание технических систем распознавания образа привлекает к себе всё большее внимание.

**Растровый метод распознавания текста**

Если исходный документ имеет типографское качество (достаточно крупный шрифт, отсутствие плохо напечатанных символов или исправлений), то задача распознавания решается методом сравнения с растровым шаблоном.

-Сначала растровое изображение страницы разделяется на изображения отдельных символов.

-Затем каждый из них последовательно накладывается на шаблоны символов, имеющихся в памяти системы, и выбирается шаблон с наименьшим количеством точек, отличных от входного изображения.

# РАСЧЁТНАЯ ЧАСТЬ

## Выбор метода распознавания текста

Под распознаванием текста обычно понимают три главных метода:

* Сравнение с заранее подготовленным шаблоном.
* Распознавание с использованием критериев, распознаваемого объекта.
* Распознавание при помощи самообучающихся алгоритмов, в том числе при помощи нейронных сетей.

Для оптического распознавания образов можно применить метод перебора вида объекта под различными углами, масштабами, смещениями и т. д. Для букв нужно перебирать шрифт, свойства шрифта и т. д.

Второй подход — найти контур объекта и исследовать его свойства (связность, наличие углов и т. д.)

Ещё один подход — использовать искусственные нейронные сети. Этот метод требует либо большого количества примеров задачи распознавания (с правильными ответами), либо специальной структуры нейронной сети, учитывающей специфику данной задачи.

Чтобы выбрать оптимальный для реализации задачи данной работы метод распознавания текста, необходимо проанализировать следующие параметры: расположение элементов текста, используемый шрифт, диапазон используемых символов.

Согласно Приложению 2 к «Инструкции по заполнению форм документов индивидуального (персонифицированного) учета в системе обязательного пенсионного страхования», заполнение персональных данных на лицевой стороне страхового свидетельства обязательного пенсионного страхования (ССОПС) осуществляется по форме АДИ-7 (см. рисунок 2.1).

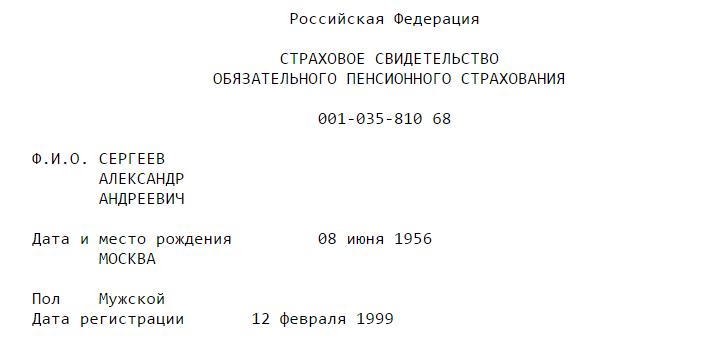


Рисунок . – Образец заполнения формы АДИ-7.

Согласно данному образцу заполнение формы персональными данными производится в строго фиксированных координатах изображения документа относительно положения надписей бланка («Ф. И. О.», «Пол» и т.п.). Согласно проведённому мной анализу изображений, имеет место незначительное отклонение от этих координат. Из данных фактов следует, что при распознавании текста из изображения страхового свидетельства необходимо определить координаты и габариты полей данных внутри изображения и после этого распознать текст значения каждого параметра.

Предположительно, надписи исполняются стандартным шрифтом Arial, ввиду высокой степени их визуальной идентичности этому шрифту. Общедоступная нормативная документация не содержит информацию о шрифте, используемом при заполнении формы АДИ-7, однако аналогичный шрифт встречается на всех проанализированных изображениях. Поэтому в рамках данной курсовой работы примем шрифт Arial единственным шрифтом для распознавания текстовых данных в ССОПС.

Проведём анализ используемых символов. В персональных данных на страховом свидетельстве используются следующие символы:

- цифры (в номере СНИЛС, номере года и числе месяца);

- заглавные буквы русского алфавита (в фамилии, имени, отчестве);

- строчные буквы русского алфавита (в названии месяца, в слове «года», в значении параметра «пол»);

- знак дефиса (в номере СНИЛС в качестве разделителя групп цифр, в некоторых фамилиях, именах и отчествах).

Вывод: В качестве оптимального метода распознавания текста, содержащегося в изображении ССОПС, применим метод с использованием шаблонов. В пользу этого метода выступают простота его реализации, а также стандартизация параметров распознаваемого текста (одно и то же расположение надписей, один и тот же шрифт).

## Предварительная обработка изображения

Чтобы в наибольшей мере исключить влияние других графических элементов образа (фона, водяных знаков) на процесс распознавания, производят процесс порогового преобразования, бимодальной сегментации (бинаризации) исходного изображения, или, другими словами, приведения его к монохромному виду. При этом каждый пиксель изображения будет обладать либо чёрным цветом («цветом текста»), либо белым («цветом фона»).

Пиксели, цвет которых с определённым максимальным отклонением совпадает с цветом текста, необходимого для извлечения, преобразуются в чёрные, остальные пиксели – в белые. Текст документа ССОПС обладает чёрным цветом. Это значит, что после бинаризации текст продолжит обладать чёрным цветом, а весь фон приобретёт белый цвет.

Для начала изображение открывается в режиме оттенков серого. Затем производится бимодальная сегментация пикселей на основе оптимального порогового значения , имеющего значение от 0 до 255.

Бинаризация с нижним порогом является наиболее простой операцией, в которой используется только одно значение порога, и может быть реализована согласно правилу:

где – яркость пикселя на исходном изображении,

– значение пикселя результирующего изображения,

– порог бинаризации.

Данная операция преобразует полутоновое изображение в бинарное. Любая точка изображения, для которой выполняется условие называется точкой объекта, а в противном случае – точкой фона.

Основная проблема операции порогового преобразования заключается в выборе надлежащего значения порога. Определение оптимального порога при преобразовании изображений является важной и трудной задачей, и для ее решения разработано много различных методов.

Очевидный способ выделения объектов из окружающего фона состоит в выборе значения порога t , разграничивающего моды распределения яркостей на изображении. Простейший из методов пороговой обработки состоит в разделении гистограммы изображения на две части с помощью единого глобального порога. После этого сегментация изображения осуществляется путем поэлементного сканирования изображения, при этом каждый пиксель отмечается как относящийся к объекту или фону, в зависимости от того, превышает ли яркость данного пикселя значение порога t или нет. Успешность этого метода целиком зависит от того, насколько хорошо гистограмма изображения поддается разделению (см. рисунок 2.2).

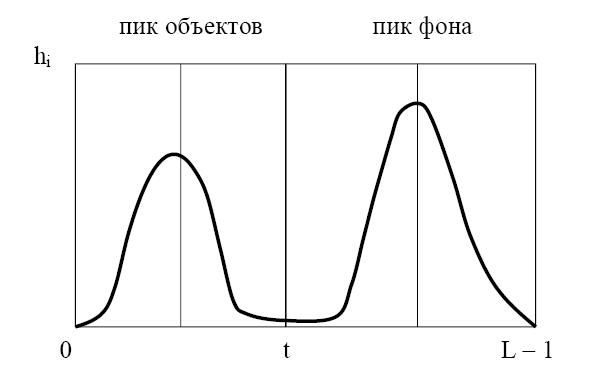
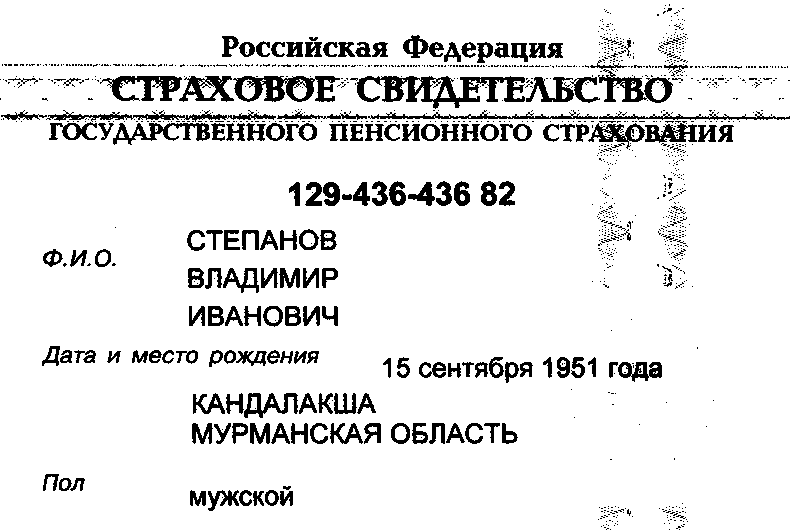
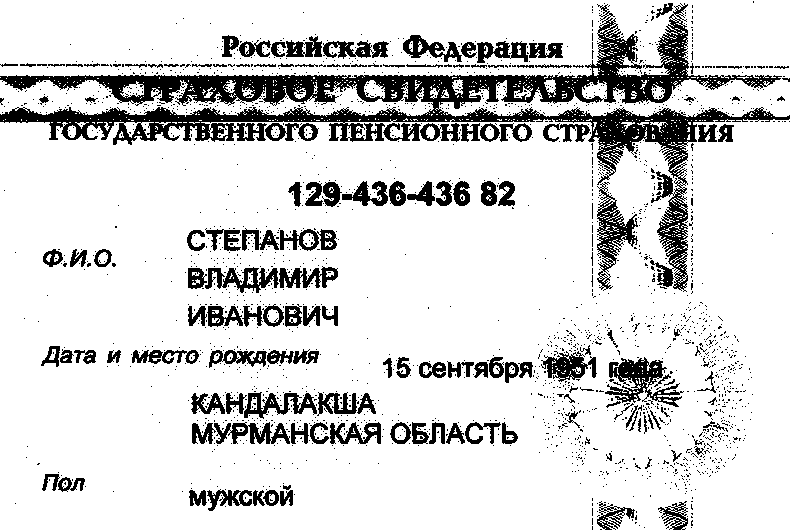
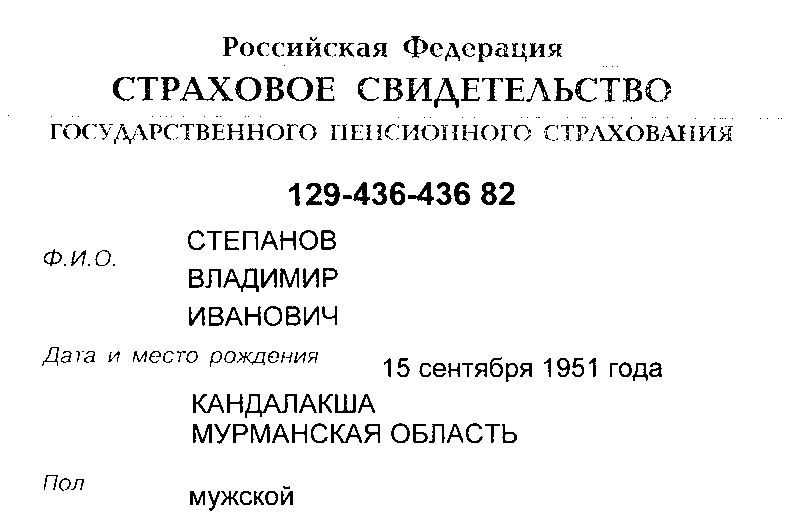
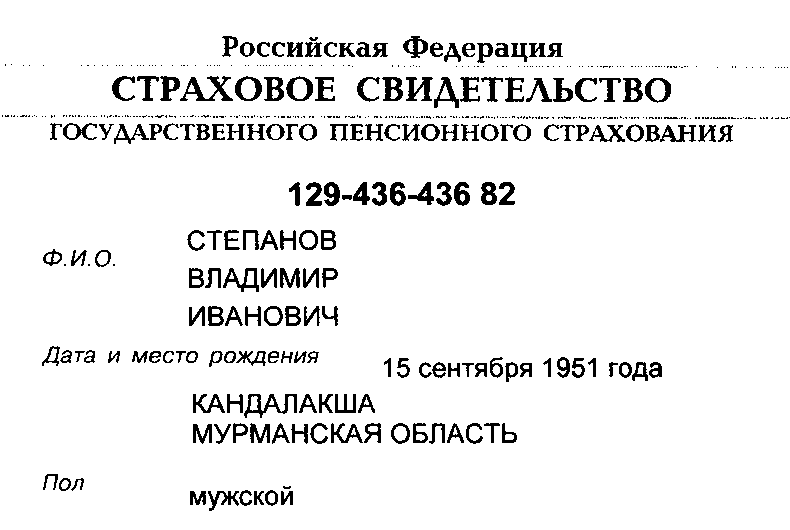


Рисунок . – Впадина между пиками гистограммы

Определение величины порога с помощью гистограммы яркостей является простым методом, который позволяет достичь «чистой» сегментации, если гистограмма изображения носит четко выраженный бимодальный характер. Такая форма гистограммы означает, что на изображении можно различить два вида сравнительно часто встречающихся пикселей – яркие и темные. При этом гистограмма легко разделяется с помощью одиночного глобального порога , расположенного во впадине между пиками гистограммы. Результаты пороговой обработки документа при разных значениях порога *t* показаны на рисунках ниже.



а) б)



в) г)

Рисунок . – Бинаризация при разных значениях порога.  
а) при *t* = 192, б) при *t* = 160, в) при *t* = 128, г) при *t* = 96

В виду различных цветовых и яркостных параметров изображений очевидна необходимость автоматизации выбора оптимального порогового значения.

Для автоматического выбора значения порога в данной курсовой работе применяется следующий итеративный алгоритм:

1. Выбирается некоторая начальная оценка значения порога .

2. Выполняется сегментация изображения с помощью порога . В результате образуются две группы пикселей: , состоящая из пикселей с яркостью больше , и , состоящая из пикселей с яркостью меньше или равной .

3. Вычисляются средние значения и яркостей пикселей по областям и соответственно.

4. Вычисляется новое значение порога:

5. Повторяются шаги со 2-го по 4-й до тех пор, пока разница значений порога в соседних итерациях не окажется меньше наперед заданного параметра .

Параметр *ε*используется для остановки алгоритма, когда изменения на каждой итерации становятся малы по сравнению с заданным параметром. Данные меры применены из соображений скорости вычислений.

## Поиск сегментов связности

Подготовленное изображение попадает на вход модуля сегментации. Задачей этого модуля является выявление структурных единиц текста – строк, слов и символов. Выделение фрагментов высоких уровней, таких как строки и слова, может быть осуществлено на основе анализа промежутков между тёмными областями.

К сожалению, такой подход не может быть применён для выделения отдельных букв, поскольку, в силу особенностей начертания или искажений, изображения соседних букв могут объединяться в одну компоненту связанности или наоборот — изображение одной буквы может распадаться на отдельные компоненты связанности. Во многих случаях для решения задачи сегментации на уровне букв используются сложные эвристические алгоритмы.

Выявление отдельных чёрных фигур производится методом заливки. Создаётся матрица, параллельная матрице пикселей обрабатываемого изображения и равная ей по размерам. Эта матрица заполняется значениями номеров сегментов, начиная от 0, в ячейках, где пиксель обладает чёрным цветом, и значением -1, если ячейка соответствует белому пикселю. Каждый пиксель бинаризированного изображения анализируется, и если рассматриваемый в данный момент пиксель принадлежит текстовой группе пикселей, ему присваивается новый номер. Затем этот же номер рекурсивно присваивается всем смежным чёрным пикселям.

После определения крайних координат (прямоугольных рамок) каждой формы из обособленной группы чёрных пикселей совершается попытка разбиения сегментов на несколько символов, если ширина сегментов превышает некоторый допустимый порог. В данной курсовой работе учитываются следующие пропорции сегментов: ширина, по возможности, не должна превышать 0,9 от высоты сегмента. При попытке отделения очередного символа от группы символов, склеенных между собой по причине низкого качества сканирования, производится поиск вертикального среза в диапазоне координаты x с 0,4 высоты сегмента по 1,4 высоты сегмента. В случае нахождения среза с минимальным количеством черных пикселей в количестве не более 4-х производится отделение по этому срезу одного сегмента на два. Процесс повторяется вплоть до достижения всех частей исходного сегмента оптимальных пропорций.

При распределении (группировке) сегментов по строкам текста производится сначала сортировка по оси Y, а затем отделение их друг от друга по признаку значительного вертикального интервала (разница в координатах нижних частей двух символов больше, чем высота символа).

Сортировка символов внутри строк в порядке следования слева направо производится по оси X – по координате левого края рамки каждого сегмента в строке.

В виду указанных особенностей работы программы необходимо учитывать, что расположение документа при сканировании должно быть горизонтальным не перевёрнутым. Допускается некоторое угловое смещение от строгой горизонтальной ориентации документа. При достижении определённого угла наклона документа при сканировании происходит накладывание строк текста друг на друга в проекции по оси Y, что значительно затрудняет построчное разделение текста.

## Сопоставление сегмента с шаблонами символов

Основная проблема распознавания текста с помощью шаблонов состоит в собирании изображения из заготовленных образов, которое станет идентично исходному изображению.

Изображения всех символов, которые практически используются при заполнении формы АДИ-7, исполнены с использованием шрифта Arial 100-го кегля и включены в программный проект данной курсовой работы в качестве электронных изображений в формате BMP (см. рисунок 2.3).



Рисунок . – Шаблоны символов.

Суть сопоставления сегмента с шаблонами состоит в поочередном сравнении двух соответствующих пикселей двух черно-белых картинок. Поскольку два пикселя должны быть соответствующими, то картинки должны быть одинаковы по размеру. Для этого сегмент растягивается до размеров текущего шаблона.

Разработана функция imgcmp(img1, img2), которая возвращает результат сравнения двух изображений в виде значения от 0 (полная инверсия) до 1 (полное соответствие). Для вычисления уровня идентичности используется функция ImgCrop.difference(image1, image2) библиотеки PIL, которая возвращает изображение, являющееся следствием разностного преобразования двух исходных. Данное результирующее изображение переводится в бинаризированный вид и подсчитывается количество черных пикселей в нём, которые свидетельствуют о совпадении в их местах оттенков цвета у исходных изображений. Отношение найденного количества к общему количеству пикселей в изображении (доля чёрных пикселей) возвращает коэффициент идентичности сегмента с шаблоном.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## Минимальные требования

Для работы спроектированной программы в ОС Windows необходима установка следующих программ и компонентов:

* Среда разработки и выполнения Python IDLE 2.7.12 (Ссылка: [https://www.python.org/downloads/release/python-2712/](https://vk.com/away.php?to=https%3A%2F%2Fwww.python.org%2Fdownloads%2Frelease%2Fpython-2712%2F)).
* Библиотека Python Image Library (PIL) 1.1.7 (Ссылка для установки в ОС Windows: [http://www.pythonware.com/products/pil/](https://vk.com/away.php?to=http%3A%2F%2Fwww.pythonware.com%2Fproducts%2Fpil%2F)).

Для запуска программы под управлением ОС Linux (или Ubuntu) по умолчанию нет необходимости в установке интерпретатора Python. Для установки библиотеки PIL потребуются следующие действия:

* Запустить Терминал
* Последовательно ввести команды:

sudo apt-get install libjpeg libjpeg-dev libfreetype6 libfreetype6-dev zlib1g-dev

ln -s /usr/lib/x86\_64-linux-gnu/libjpeg.so /usr/lib

ln -s /usr/lib/x86\_64-linux-gnu/libfreetype.so /usr/lib

ln -s /usr/lib/x86\_64-linux-gnu/libz.so /usr/lib

pip install -U PIL

## Инструкция по использованию

1. Запустить файл SNILS.py
2. После появления предложения для ввода ввести полное имя файла с изображением документа СНИЛС.
3. Через некоторое время на экране появится информация об извлечённых текстовых данных.
4. Для продолжения работы с программой перейти к пункту 2.
5. Закрытие окна программы.

# ЗАКЛЮЧЕНИЕ

К недостаткам представленного в данной курсовой работе решения можно отнести следующие заключения:

— Программа не может распознавать буквы, состоящие из двух фигур, например, Ы, Й.

— При повороте/шумах/прочем распознавание не корректно.

— Иногда программа распознает строчные символы как прописные, и наоборот.

Достоинства использованных методов и алгоритмов состоят в низкой сложности разработки

# ПРИЛОЖЕНИЕ

**Листинг программного кода**

#!/usr/bin/env python

# -\*- coding: cp1251 -\*-

from PIL import Image, ImageChops, ImageDraw

import time

# -------------------------------------------------------------------------------

def binarize(img):

''' Возвращает черно-белую копию изображения. '''

# Инициализация переменных.

imgRes = img.convert('L') # Копирование исходного изображения.

pix = img.load() # Матрица исходных пикселей.

pix2 = imgRes.load() # Матрица черно-белых пикселей.

histogram = img.histogram() # Гистограмма исходного изображения.

oldThreshold = 0 # Предыдущее пороговое значение.

threshold = 127 # Текущее пороговое значение.

maxCorrelation = 5 # Наибольшая разница между значениями порога.

# Выбор оптимального порогового значения.

while abs(threshold - oldThreshold) > maxCorrelation:

# Параметры текущего разделения на фон и текст.

numBackPixels = 0 # Количество пикселей фона.

numTextPixels = 0 # Количество пикселей текста.

sumBackPixels = 0 # Сумма значений пикселей фона.

sumTextPixels = 0 # Сумма значений пикселей текста.

# Сегментация по текущему пороговому значению.

for val in range(256):

numVal = histogram[val] # Количество пикселей с оттенком val.

# Определение параметров текущего разделения на фон и текст.

if (val < threshold):

# Количество и сумма значений пикселей текста.

numTextPixels = numTextPixels + 1

sumTextPixels = sumTextPixels + val

else:

# Количество и сумма значений пикселей фона.

numBackPixels = numBackPixels + 1

sumBackPixels = sumBackPixels + val

# Расчет нового порогового значения.

oldThreshold = threshold

threshold = ((sumBackPixels/numBackPixels) +

(sumTextPixels/numTextPixels)) / 2

# Бимодальное пороговое преобразование изображения.

threshold = threshold + maxCorrelation

for i in range(img.size[0]):

for j in range(img.size[1]):

# Строгое присвоение пикселю черного или белого цвета.

if pix[i,j] < threshold:

pix2[i,j] = 0

else:

pix2[i,j] = 255

return imgRes # Вернуть полученное изображение.

# -------------------------------------------------------------------------------

def check\_near(x, y, group):

''' Находит сегменты связности из смежных черных пикселей. '''

pixGroups[x][y] = group # Присвоить пикселю № группы.

# Цикл по соседним пикселям.

for near in nears:

x2 = x + near[0] # Абсолютные координаты

y2 = y + near[1] # соседнего пикселя.

# Исключить координаты, выходящие за край изображения.

if x2 < 0 or x2 >= imgBin.size[0]:

continue

if y2 < 0 or y2 >= imgBin.size[1]:

continue

# Если соседний пиксель - чёрный, и не входит в группу текущего.

if (pix[x2, y2] == 0 and pixGroups[x2][y2] <> group):

# Если сосед находится в группе с другим номером.

if pixGroups[x2][y2] <> -1 and pixGroups[x2][y2] <> group:

# то присваивать соседам меньший из двух номеров.

if pixGroups[x2][y2] < group:

group = pixGroups[x2][y2]

else:

pixGroups[x2][y2] = group

# Расширение рамки сегмента.

if bounds[group][0] > x2:

bounds[group][0] = x2

elif bounds[group][2] < x2:

bounds[group][2] = x2

if bounds[group][1] > y2:

bounds[group][1] = y2

elif bounds[group][3] < y2:

bounds[group][3] = y2

global depth # Определить текущий уровень рекурсии.

depth = depth + 1 # Новый уровень рекурсии.

if depth < 500: # Если глубина рекурсии не превышает 500,

check\_near(x2, y2, group) # присвоить соседу тот же № группы.

depth = depth - 1 # Возврат на предыдущий уровень рекурсии.

# -------------------------------------------------------------------------------

def imgcmp(img1, img2):

''' Возвращает коэффициент идентичности двух изображений. '''

width = img2.size[0] # Ширина шаблона.

height = img2.size[1] # Высота шаблона.

img1 = img1.resize((width, height)) # Подогнать вырезку под размеры шаблона.

dif = ImageChops.difference(img1, img2) # Разностное преобразование.

dif = dif.convert('1') # Бинаризация.

# Вернуть долю черных пикселей.

return float(dif.histogram()[0]) / (width \* height)

# -------------------------------------------------------------------------------

def print\_field(label, left, top, right, bottom):

''' Отображает распознанный текст, заключенный в указанных границах поля. '''

data = '' # Строки значения поля.

couples = [] # Массив сегментов в виде пар [символ, рамка].

past = None # Рамка предыдущего символа.

for b in range(len(bounds)):

bound = bounds[b] # Рамка текущего сегмента.

# Если сегмент полностью в указанных границах и достаточно большой.

if (bound[0] > imgBin.size[0] \* left and

bound[1] > imgBin.size[1] \* top and

bound[2] < imgBin.size[0] \* right and

bound[3] < imgBin.size[1] \* bottom and

text[b] <> '.'):

couples.append([text[b], bound]) # Добавить в массив сегментов.

# Группировка символов по строкам и сортировка внутри строк.

couples.sort(key=sortByY) # Сортировать по вертикальному расположению.

first = 0 # Индекс первого сегмента в строке.

for c in range(len(couples)):

couple = couples[c] # Текущий сегмент.

if c: # Если это не первый символ массива.

# Если между символами существенная разница по оси Y

# Или символ последний из массива.

if (couple[1][3] - past[3] > past[3] - past[1] or

c == len(couples)-1):

srez = couples[first:c]

srez.sort(key=sortByX) # Сортировать символы в строке по X.

couples[first:c] = srez

# Добавить знак-индикатор переноса строк.

couples[c-1][0] = couples[c-1][0] + ('|'

if c < (len(couples)-1) else '')

first = c # Определить индекс первого символа след. строки.

past = couple[1] # Определить текущую рамку как предыдущую.

# Сформировать строку значения поля.

for c in range(len(couples)):

couple = couples[c]

data = data + couple[0] # Конкатенация очередного символа.

if c: # Если это не первый символ массива.

# Если есть существенный разрыв по оси X,

if couple[1][0] - past[2] > (past[2] - past[0]) / 2:

couple[0] = ' ' + couple[0] # определить его как пробел.

past = couple[1] # Определить текущую рамку как предыдущую.

print (label + ': ' + data) # Вывести на экран значение поля.

# Прорисовка прямоугольных границ поля.

drawRects.rectangle((imgBin.size[0] \* left, imgBin.size[1] \* top,

imgBin.size[0] \* right, imgBin.size[1] \* bottom),

fill=None, outline='#00FF00')

# -------------------------------------------------------------------------------

def split\_segment(b):

''' Разделяет горизонтальный сегмент на символы. '''

global bounds # Определить текущий массив границ сегментов.

width = float(b[2] - b[0]) # Ширина сегмента.

height = float(b[3] - b[1]) # Высота сегмента.

# Цикл разделения символа, пока ширина сегмента превышает 0,9 от его высоты.

while (width > height \* 0.9):

v\_min = height # Минимальное количество черных пикселей.

x\_min = b[0] + int(height \* 0.8) # Значение x разреза с v\_min.

# Поиск вертикального разреза с минимальным кол-вом пикселей текста.

for x in range(b[0] + int(height \* 0.35), b[0] + int(height \* 1.4)):

if x >= b[2]:

break # Завершить, если достигнут правый край.

v = 0 # Счетчик черных пикселей.

# Цикл по пикселям разреза.

for y in range(b[1], b[3]+1):

if (pix[x,y] == 0):

v = v + 1 # Найден ещё один чёрный пиксель.

if (v\_min > v): # Запомнить срез, если пока он самый оптимальный.

v\_min = v

x\_min = x

if (v\_min <= 3):

bounds.append([b[0], b[1], x\_min-1, b[3]]) # Отрезание сегмента.

b[0] = x\_min + 1 # Пересчет рамок исходного сегмента.

width = float(b[2] - b[0]) # Пересчёт ширины сегмента.

else:

break # Пропустить нарезку, если в срезе больше 3-х пкс текста.

# -------------------------------------------------------------------------------

def sortByX(c): # Сортировка сегментов-символов по x внутри строки.

return c[1][0]

def sortByY(c): # Сортировка сегментов-символов по y внутри поля.

return c[1][3]

# ===============================================================================

while 1:

# --- ОТКРЫТИЕ И ПРЕОБРАЗОВАНИЕ ИЗОБРАЖЕНИЯ -------------------------------------

imgOrig = Image.open(raw\_input('Введите полное имя файла СНИЛС: ')) # Открыть.

imgOrig.draft('L', imgOrig.size) # Приведение изображения к оттенкам серого.

imgBin = binarize(imgOrig) # Бимодальная сегментация.

# --- ОТКРЫТИЕ КОЛЛЕЦИИ ШАБЛОНОВ ------------------------------------------------

# Массив имён файлов-шаблонов (\*.bmp).

characters = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0',

"А", "Б", "В", "Г", "Д", "Е", "Ё", "Ж",

"З", "И", "Й", "К", "Л", "М", "Н", "О",

"П", "Р", "С", "Т", "У", "Ф", "Х", "Ц",

"Ч", "Ш", "Щ", "Ъ", "Ы", "Ь", "Э", "Ю", "Я",

"а-", "б-", "в-", "г-", "д-", "е-", "ё-", "ж-",

"з-", "и-", "й-", "к-", "л-", "м-", "н-", "о-",

"п-", "р-", "с-", "т-", "у-", "ф-", "х-", "ц-",

"ч-", "ш-", "щ-", "ъ-", "ы-", "ь-", "э-", "ю-", "я-"]

listChars = [] # Массив изображений шаблонов.

# Поочередное открытие шаблонов символов в оттенках серого.

for iChar in range(len(characters)):

char = characters[iChar]

listChars.append(Image.open('patterns\\' + char + '.bmp'))

listChars[-1].draft('L', listChars[-1].size)

characters[iChar] = char.replace('-', '')

# --- СЕГМЕНТАЦИЯ ИЗОБРАЖЕНИЯ ---------------------------------------------------

group = -1 # Номер группы пикселей.

bounds = [] # Границы рамок групп пикселей.

pix = imgBin.load() # Матрица пикселей бинаризированного изображения.

pixGroups = [] # Матрица принадлежности пикселей к группам.

for x in range(0, imgBin.size[0]):

pixGroups.append([])

for y in range(0, imgBin.size[1]):

pixGroups[x].append(-1)

# Массив относительных координат смежных пикселей.

nears = [[-1, 0],

[-1, -1],

[0, -1],

[1, -1],

[1, 0],

[1, 1],

[0, 1],

[-1, 1]]

depth = 0

# Разделение пикселей на сегменты связности.

for y in range(imgBin.size[1]):

for x in range(imgBin.size[0]):

# Если пиксель черный и ему не присвоена группа.

if (pix[x, y] == 0 and pixGroups[x][y] == -1):

depth = 0 # Инициализация значения глубины рекурсии.

group = group + 1 # Увеличить счётчик групп.

bounds.append([x, y, x, y]) # Начальные границы группы.

# Присвоить новый № группы текущему и соседним пикселям.

check\_near(x, y, group)

# Обработка и прорисовка сегментов.

for b in bounds:

width = float(b[2] - b[0]) # Ширина сегмента.

height = float(b[3] - b[1]) # Высота сегмента.

if (width < 3 or height < 5):

continue # Пропустить слишком маленький сегмент.

if (width > height \* 0.9):

split\_segment(b) # Разделить склеенные символы.

# --- СОПОСТАВЛЕНИЕ СЕГМЕНТОВ С ШАБЛОНАМИ ---------------------------------------

minIdentityValue = 0.65 # Минимально допустимое значение идентичности.

imgWhite = ImageChops.constant(imgDraw, 255) # Отладочное полотно № 2.

text = "" # Строка с распознанными символами.

# Цикл по сегментам.

for b in bounds:

# Пропускать слишком маленькие сегменты.

if (b[2]-b[0] < 3 or b[3]-b[1] < 5):

text = text + '.'

continue

# Обрезка сегмента в отдельное изображение.

imgCrop = imgBin.crop((b[0],b[1],b[2]+1,b[3]+1))

# Сравнить сегмент с каждым шаблоном.

matchValue = 0.0 # Наибольшее из найденных значение идентичности.

for iChar in range(0, len(listChars)):

imgChar = listChars[iChar] # Изображения символа-шаблона.

identity = imgcmp(imgCrop, imgChar) # Идентичность шаблону.

# Если идентичность больше допустимой и ранее найденной.

if (identity > minIdentityValue and identity > matchValue):

# Запомнить текущий шаблон.

matchValue = identity

matchChar = iChar

# Пропустить отладочную прорисовку, если похожий символ не найден.

if (matchValue == 0.0):

text = text + '?'

continue

text = text + characters[matchChar]

# --- ВЫВОД НЕОБХОДИМЫХ ПОЛЕЙ ---------------------------------------------------

print\_field('№ СНИЛС', 100.0/700, 140.0/500, 600.0/700, 185.0/500)

print\_field('Ф.И.О.', 150.0/700, 175.0/500, 600.0/700, 290.0/500)

print\_field('Дата рождения', 300.0/700, 230.0/500, 600.0/700, 300.0/500)

print\_field('Место рождения', 100.0/700, 285.0/500, 600.0/700, 360.0/500)

print\_field('Пол', 100.0/700, 360.0/500, 500.0/700, 410.0/500)

print\_field('Дата регистрации', 200.0/700, 400.0/500, 600.0/700, 480.0/500)

print ('')

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методы, алгоритмы и программное обеспечение гибких информационных технологий для автоматизированных идентификационных систем: Сб. научн. ст. - Мн.: БГУ. - 1999. - 182 с.
2. Селянинов М.Ю., Чернявский Ю.А. Сегментация дактилоскопических изображений в автоматизированных информационных системах // Информатика - 2005. - №2. - С. 86 - 92.
3. Р. Гонсалес, Р. Вудс. Цифровая обработка изображений: Пер. с англ. - М.: Техносфера, 2005. - 1072 с.
4. Абламейко С.В., Лагуновский Д.М. Обработка изображений: технология, методы, применение. Учебное пособие. - Мн.: Амалфея, 2000. - 304 с.
5. Методы компьютерной обработки изображений / Ред. Сойфер В.А. - М.: Физматлит, 2001. - 784 с.
6. Цифровая обработка изображений в информационных системах: Учебное пособие / И.С. Грузман, В.С. Киричук и др. - Новосибирск, НГТУ, 2002. - 352 с.
7. Квасников В.П., Дзюбаненко А.В. Улучшение визуального качества цифрового изображения путем поэлементного преобразования // Авиационно-космическая техника и технология 2009 г., 8, стр. 200-204
8. Арлазаров В.Л., Куратов П.А., Славин О.А. Распознавание строк печатных текстов // Сб. трудов ИСА РАН «Методы и средства работы с документами». — М.: Эдиториал УРСС, 2000. — С. 31-51.
9. Проект СПбГУ Открытый код: распознавание текстовых изображений [Электронный ресурс] — Режим доступа: ocr.apmath.spbu.ru/
10. Багрова И. А., Грицай А. А., Сорокин С. В., Пономарев С. А., Сытник Д. А. Выбор признаков для распознавания печатных кириллических символов // Вестник Тверского Государственного Университета 2010 г., 28, стр. 59-73
11. Панфилов С. А. Методы и программный комплекс моделирования алгоритмов управления нелинейными динамическими системами на основе мягких вычислений. Диссертация на соискание ученой степени кандидата технических наук. Тверь, 2005.