Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой,
_____ К.А. Домбровский
«___»_____ 2017 г.

Обработка изображений для приемной комиссии ЮУрГУ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ– 09.03.01.2017.382.ПЗ ВКР

Руководитель работы,
к.т.н., доцент каф. «Электронные вычислительные машины»
_____ Е.С. Ярош
«___»_____ 2017 г.

Автор работы
студенты группы КЭ-445
_____В.С. Горобченко
«___»_____ 2017 г.

Нормоконтролёр, ст. преп. каф. «Электронные вычислительные машины»

_____ В.В. Лурье
«___»_____ 2017 г.

Челябинск 2017

АННОТАЦИЯ

Горобченко В.С. Обработка изображений для приемной комиссии ЮУрГУ. – Челябинск: ФГБОУ ВПО «ЮУрГУ» (НИУ) ВШЭКН; 2017, 43 с., 22 ил., 3 табл., библиогр. список – 15 наим.

Выпускная квалификационная работа выполнена с целью реализации библиотеки обработки изображений для приемной комиссии ЮУрГУ.

Были рассмотрены аналоги и альтернативные решения разрабатываемой библиотеки, проанализированы типовые дефекты и определен необходимый функционал дли их устранения.

В ходе выполнения выпускной квалификационной работы были спроектированы диаграмма классов и диаграмма вариантов использования, реализованы функции для определения положения документа по его типу, корректировки угла наклона, удаления фона от сканера, сжатия разрешения изображения и определения формата изображения.

Данная работа является актуальной, так как позволяет обработать изображения отсканированных документов, приводя их к единому, стандартному виду и подготавливая для последующего хранения и обработки сотрудниками ЮУрГУ.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время в большинстве вузов РФ возможна подача документов для поступления через интернет. Это удобный и максимально оперативный способ для абитуриентов, не требующий их личного присутствия. В ЮУрГУ такую возможность обеспечивает информационная система Универис.

Среди прочих документов абитуриенты должны предоставить в электронном виде копию документа, удостоверяющего личность (паспорта), и копию документа об образовании.

Но возникла проблема: пользователи, сканируя документы самостоятельно, загружали изображения, с которыми сотруднику ЮУрГУ неудобно работать в дальнейшем. Типовыми дефектами являются изображение, повернутое на некоторый угол, неправильная ориентация изображения, искажение формы (трапециевидное изображение) и др. Также некоторые изображения имеют большие поля, что ведет к излишним затратам памяти на их хранение. Поэтому возникла необходимость в обработке изображений, поступающих в Универис.

1. ПОСТАНОВКА И АНАЛИЗ ЗАДАЧИ

1.1. Анализ задачи

Приемная комиссия ЮУрГУ для поступления требует от абитуриента такие документы как паспорт и документ об образовании (аттестат или диплом). При подаче документов через интернет абитуриенты присылали документы, не удобные для дальнейшей работы сотрудников ЮУрГУ. Примеры изображений, требующих обработки, приведены на рисунке 1.1:

Рисунок 1.1 – Примеры изображений, требующих обработки

Документы на изображениях имеют неправильную ориентацию, что делает их неудобными для восприятия. Чтобы сделать их читаемыми, нужно привести изображение к стандартному виду (рисунок 1.2).

Рисунок 1.2 – Приведение изображения к стандартному виду

Стандартный вид предполагает возможность распознать основные параметры и реквизиты документов в соответствии с описанием, утвержденным правительством РФ. Поэтому у каждого документа имеются шаблон и основные элементы, на основании которых можно скорректировать его отсканированное изображение.

Разрабатываемый программный продукт должен, сравнивая в автоматическом режиме исходный документ с шаблоном, определять тип документа и его стандартное положение и на основе этих данных корректировать наклон, выполнять обрезку, оставляя только полезную область изображения.

Для возможности внедрения в информационную систему Универис и подключения функционала программы разрабатываемый программный продукт должен быть библиотекой.

1.2. Цель выпускной квалификационной работы

Целью выпускной квалификационной работы является разработка программного продукта, который приводит документы к единому стандартизованному виду, удобному для восприятия. В соответствии с п.1.1, можно выделить следующие требования:

- обрезка изображения;
- поворот изображения;
- определение положения документа;
- определение типа документа;
- возможность подключения функционала программы к системе Универис;
- бесплатность, открытый исходный код.

1.3. Задачи выпускной квалификационной работы

Для достижения поставленной цели необходимо решить следующие задачи:

1. Анализ существующих аналогов и альтернативных решений;
2. Спроектировать диаграмму классов и диаграмму вариантов использования;
3. Осуществить выбор инструментов для разработки;
4. Разработать программный код библиотеки.

2. АНАЛИЗ СУЩЕСТВУЮЩИХ РАЗРАБОТОК

С целью поиска аналогов требуемой библиотеки обработки изображений были проанализированы следующие средства:

- ABBYY FineReader;
- ABBYY PassportReader;
- Scan Tailor;
- ScanKromsator.

2.1. ABBYY FineReader

FineReader – это программа для оптического распознавания символов, имеющая широкий набор инструментов и поддерживающая большинство графических и текстовых форматов.

Программа автоматически применяет необходимые инструменты предварительной обработки для разных типов изображений. С помощью редактора изображений пользователь может вручную настроить яркость и контрастность фотографии, исправить перекос или трапециевидное искажение, убрать цифровой шум, обрезать лишние части изображения и многое другое [1]. Логотип ABBYY FineReader представлен на рисунке 2.1.

Рисунок 2.1 – Логотип ABBYY FineReader

Интерфейс приложения представлен на рисунке 2.2.

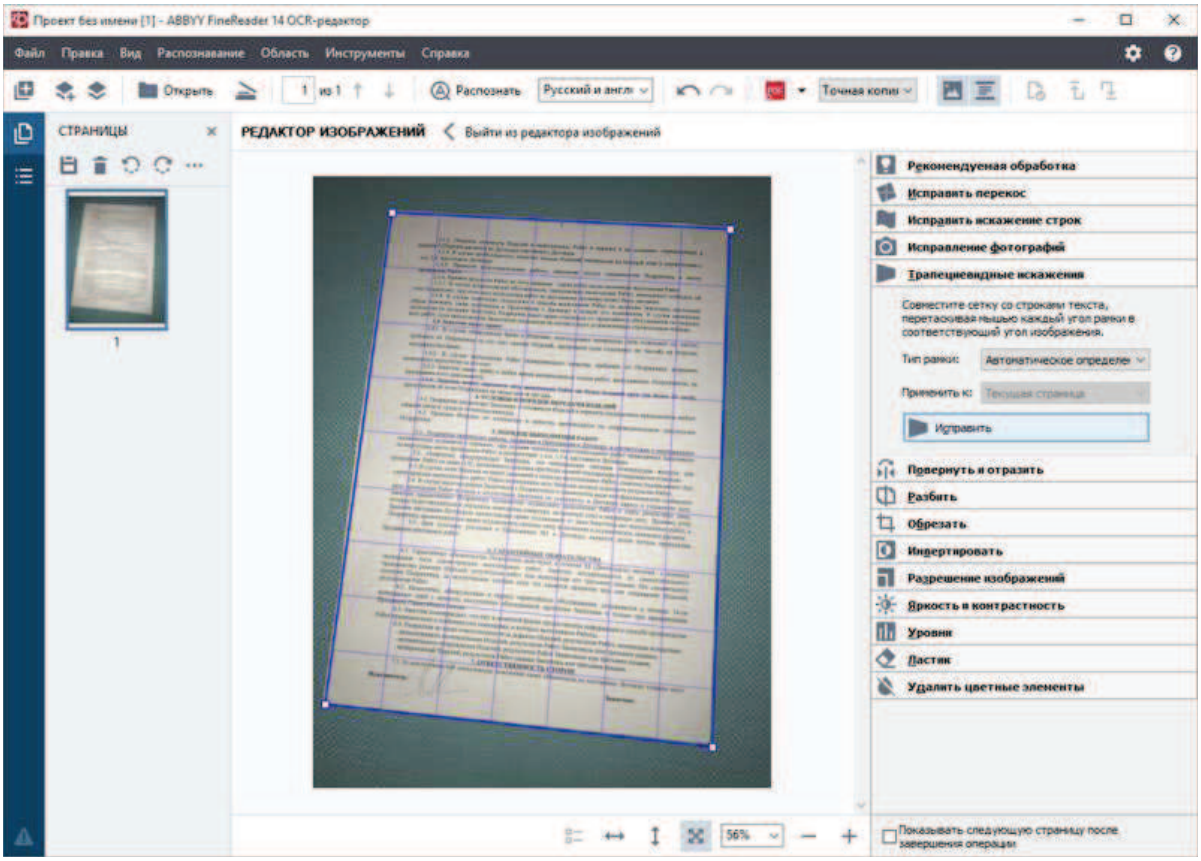

Рисунок 2.2 – Интерфейс ABBYY FineReader

Преимущества:

- поддерживаемые форматы PDF, TIFF, JPEG, JPEG 2000, JBIG2, PNG, BMP, PCX, GIF, DjVu, XPS, DOC(X), XLS(X), PPT(X), VSD(X), HTML, RTF, TXT, ODT, ODS, ODP [2];
- широкий набор инструментов, в том числе удаление фона и исправление наклона;
- предоставление API.

Недостатки:

- высокая стоимость;
- отсутствие открытого исходного кода;
- отсутствие поддержки определения паспорта и документа об образовании.

2.2. ABBYY PassportReader

Специализированный инструмент для разработчиков, который предназначен для извлечения данных из документов, удостоверяющих личность. Позволяет распознать и извлекать информацию из таких документов как паспорт, заграничный паспорт, удостоверение личности, водительское удостоверение, свидетельство о рождении. В настоящее время продукт позволяет обрабатывать оригиналы и ксерокопии документов следующих стран: Российская Федерация, Азербайджан, Белоруссия, Казахстан, Киргизия, Таджикистан, Узбекистан, Украина [3]. Логотип ABBYY PassportReader представлен на рисунке 2.3.

ABBYY PassportReader SDK

Рисунок 2.3 – Логотип ABBYY PassportReader

Интерфейс ABBYY PassportReader представлен на рисунке 2.4.

Рисунок 2.4 – Интерфейс ABBYY PassportReader

Преимущества:

- распознание паспортов восьми государств;
- широкий набор инструментов, в том числе удаление фона и исправление наклона;
- поддерживаемые форматы: IFF, PNG, JPEG, BMP;
- предоставление API.

Недостатки:

- отсутствие необходимых типов документов (документа об образовании);
- отсутствие открытого исходного кода;
- отсутствие бесплатной версии.

2.3. Scan Tailor

Scan Tailor – компьютерная программа для постобработки изображений, полученных при помощи сканера. Возможности программы [4]: исправление ориентации (поворот страниц), разрезание страниц, компенсация наклона (для горизонтального выравнивания строк), выделение полезной области, добавление полей, очистка изображений страниц. Интерфейс приложения представлен на рисунке 2.5.

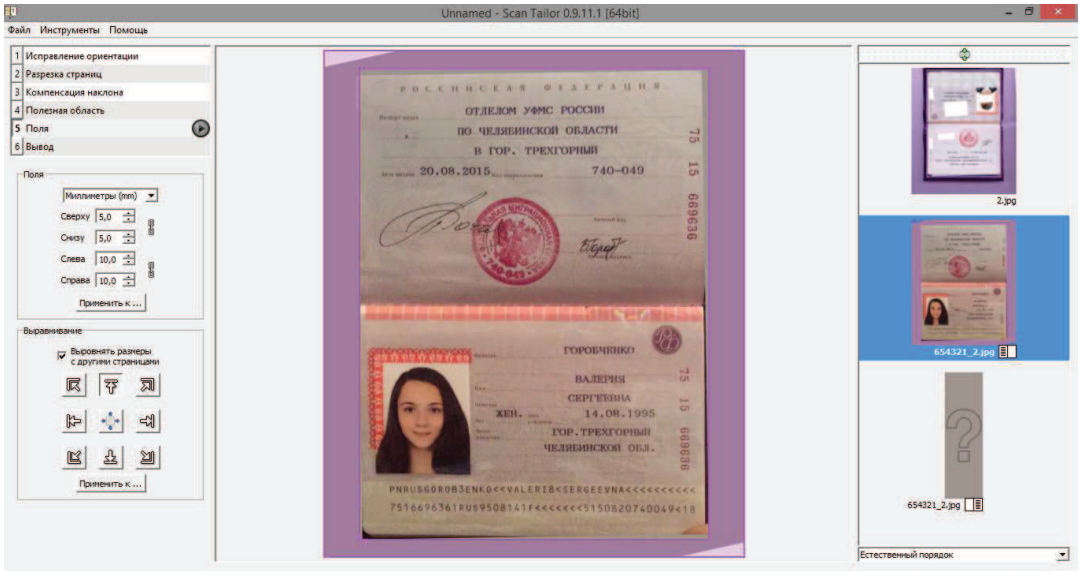

Рисунок 2.5 – Интерфейс Scan Tailor

Преимущества:

- удаление фона;
- исправление наклона;
- наличие открытого исходного кода;
- распространение программы бесплатно;

Недостатки:

- отсутствие поддержки определения паспорта и документа об образовании;
- отсутствие определения ориентации изображения по типу документа;

2.4. ScanKromsator

ScanKromsator – программа для обработки изображений, полученных при сканировании книг, журналов и другой печатной продукции. Результат обработки оптимизирован для сохранения в DjVu или PDF форматах.

Преимущества:

- удаление фона;
- исправление наклона;
- распространение программы бесплатно.

Недостатки:

- отсутствие поддержки определения паспорта и документа об образовании;
- отсутствие определения ориентации изображения по типу документа;
- отсутствие возможности подключения функционала.

2.5. Сравнительный анализ

Сравнительный анализ рассмотренных программ представлен в таблице 2.1

Таблица 2.1 – Сравнительный анализ рассмотренных программ.

Критерии	ABBYY FineReader	ABBYY PassportReader	Scan Tailor	ScanKromsator
Обрезка изображения	+	+	+	+
Поворот изображения	+	+	+	+
Определение положения документа	+	+		
Определение типа документа	+	+		
Подключение функционала программы	+	+		
Наличие бесплатной версии			+	+
Открытый исходный код			+	

Таким образом, ни одна из имеющихся программ не удовлетворяет полностью всем заданным критериям.

3. ТЕХНИЧЕСКОЕ ЗАДАНИЕ ПРОЕКТИРУЕМОЙ БИБЛИОТЕКИ

Необходимо разработать библиотеку, функциональным назначением которой является предоставление пользователю набора инструментов для приведения отсканированных документов к единому виду, удобному для восприятия.

Разрабатываемый программный продукт должен обеспечивать возможность выполнения перечисленных ниже функций:

- определение типа документа;
- определение положения документа;
- сохранение файла со сжатием разрешения изображения;
- сохранение файла в исходном разрешении;
- удаление фона от сканера;
- коррекция угла наклона документа;
- определение формата изображения.

Входными данными разрабатываемой библиотеки являются отсканированные документы в форматах JPEG, TIFF, BMP, PNG, GIF представленных в виде массива байт, а выходными данными – обработанные документы в форматах JPEG, TIFF, BMP, PNG, GIF представленных в виде массива байт.

4. ПРОЕКТИРОВАНИЕ

При разработке библиотеки обработки изображений можно выделить следующие этапы:

- построение диаграммы использования;
- построение диаграммы классов;
- выбор инструментов для разработки;
- программная реализация библиотеки обработки изображений.

4.1. Построение диаграммы использования

Диаграммы вариантов использования – основной вид диаграмм при моделировании поведения системы, подсистемы или класса. Они применяются для моделирования представления системы с точки зрения вариантов использования [6]. Диаграммы вариантов использования описывают взаимоотношения и зависимости между наборами вариантов использования и действующими лицами, участвующими в процессе. Варианты использования представляют собой действия (события), приводящие к значимому для действующего лица результату.

Желаемым результатом для пользователя проектируемой библиотеки является приведение изображения документа к стандартному виду путем корректировки наклона и удаления фона. При этом библиотека должна обрабатывать изображения в форматах JPEG, TIFF, BMP, PNG, GIF и иметь возможность выбора режима сохранения.

Построенная диаграмма использования представлена на рисунке 4.1.

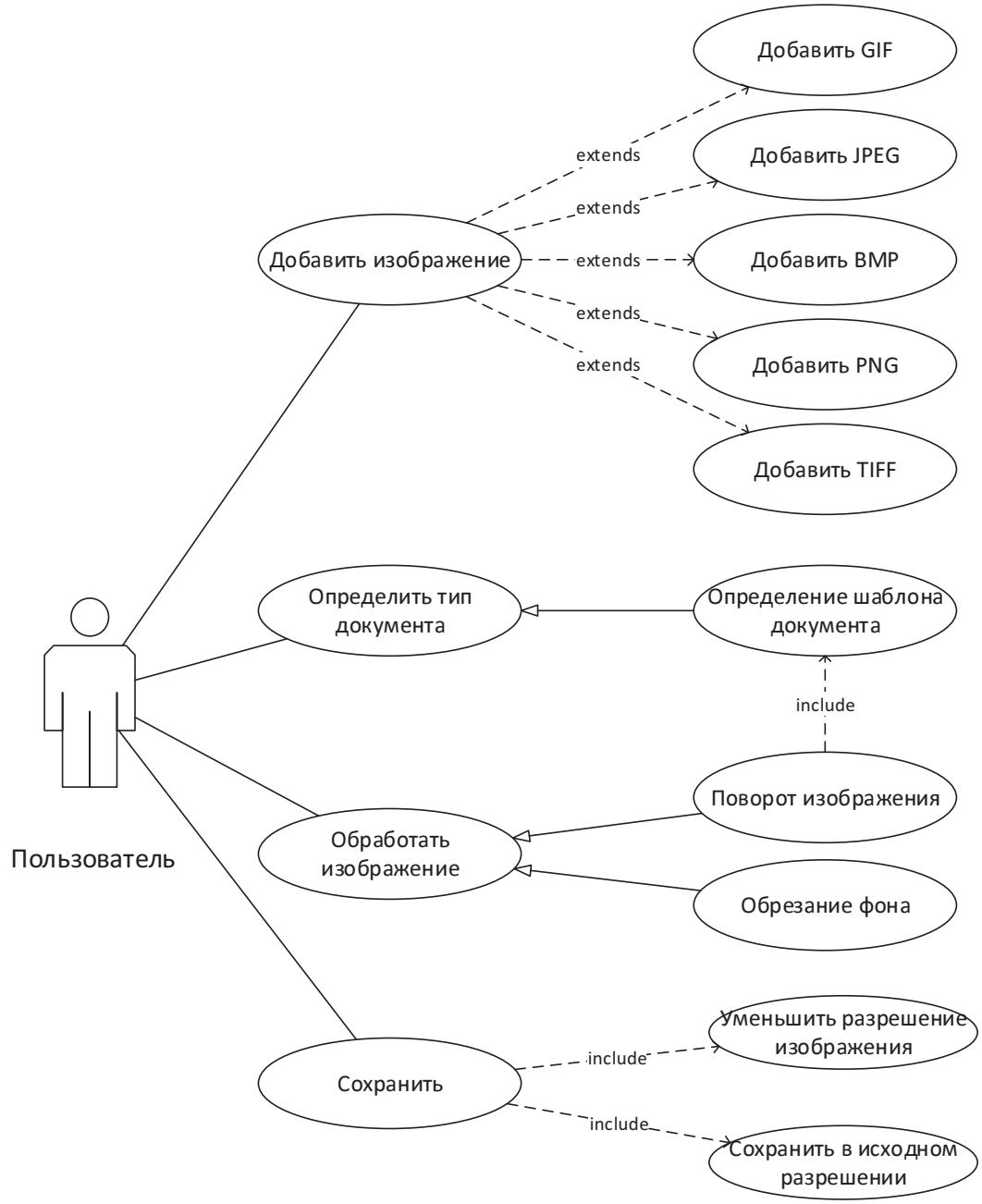

Рисунок 4.1 – Диаграмма вариантов использования

4.2. Построение диаграммы классов

Диаграммы классов – наиболее часто используемый тип диаграмм, которые создаются при моделировании объектно-ориентированных систем. Они показывают набор классов, интерфейсов и кооперации, а также их связи [6].

Построенная диаграмма классов представлена на рисунке 4.2.

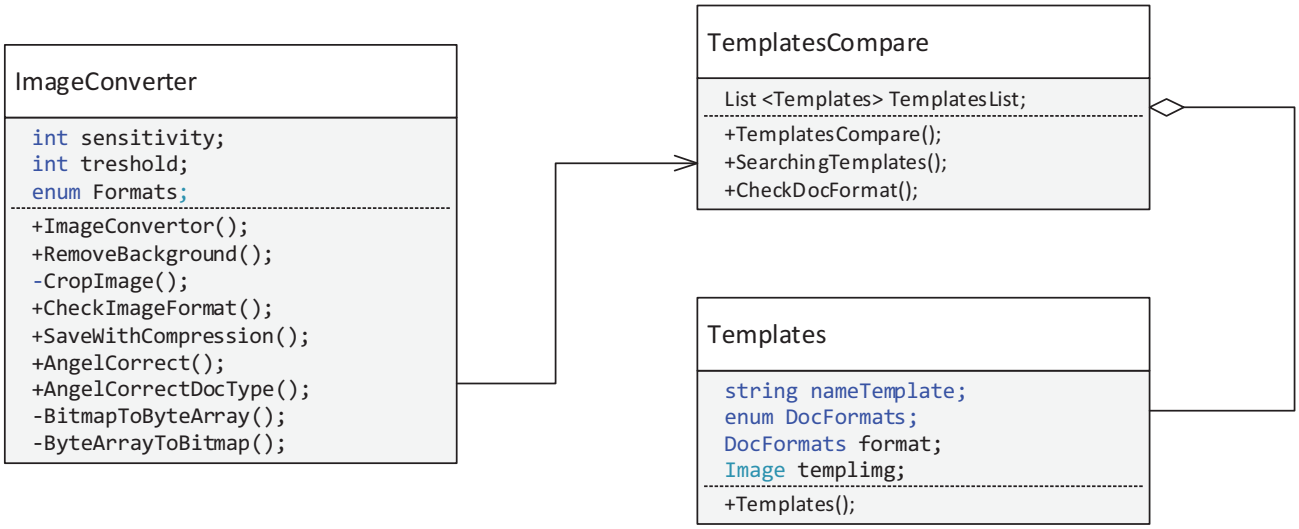

Рисунок 4.2 – Диаграмма классов

Описание классов представлено в таблице 4.1:

Таблица 4.1 – Описание классов

Название класса	Описание
ImageConverter	Основной класс проектируемой библиотеки, содержащий весь набор инструментов, реализующий функционал проектируемой библиотеки.
TemplatesCompare	Класс для работы с шаблонами.
Templates	Класс для хранения шаблонов.

Помимо классов, представленных на рисунке 4.2, необходима реализация класса для обработки исключений, возникающих при ошибке, например, невозможности обработки изображения.

Класс ImageConverter включает следующие атрибуты:

- int sensitivity – значение перепад интенсивности соседних пикселей;
- int treshold – пороговое значение количества перепадов яркости;
- public enum Formats – перечисление форматов изображения.

Описание методов класса ImageConverter представлено в таблице 4.2:

Таблица 4.2 – Описание методов класса ImageConverter

Метод	Описание
ImageConvertor ();	Конструктор класса.
RemoveBackground ();	Метод удаления фона.
CropImage();	Метод для обрезки фона.
CheckImageFormat();	Метод проверки формата изображения.
SaveWithCompression();	Метод сохранения с сжатием.
AngelCorrect();	Метод коррекции угла наклона по основным линиям.
AngelCorrectDocType();	Метод коррекции угла наклона по шаблону.
BitmapToByteArray();	Метод конвертирования Bitmap в массив байт.
ByteArrayToBitmap();	Метод конвертирования массива байт в Bitmap.

Класс TemplatesCompare содержит атрибут List<Templates> TemplatesList – список для хранения шаблонов, конструктор данного класса TemplatesCompare, метод поиска шаблона SearchingTemplates и метод определения типа документа CheckDocFormat.

Класс Templates включает в себя следующие атрибуты:

- string nameTemplate – имя шаблона;
- enum DocFormats – перечисление форматов документов;
- DocFormats format – формат шаблона;
- Image templimg – шаблон;

4.3. Выбор инструментов для разработки

4.3.1. Язык программирования

Поскольку разрабатываемый программный продукт должен предоставлять пользователю набора инструментов для обработки отсканированных документов и предполагает внедрение в уже работающую информационную систему Универис, реализованную на платформе Microsoft .NET Framework на языке программирования C#, он должен быть реализован на том же языке программирования для удобства конечного пользователя. Поэтому язык программирования является заданным.

C# – объектно-ориентированный язык программирования, разработанный в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML [11].

4.3.2. Выбор среды разработки

В качестве среды разработки была выбрана Visual Studio 2015. Данная среда позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight [7].

4.3.3. Выбор графической библиотеки

Для решения задач, связанных с обработкой изображений, требовалась выбрать графическую библиотеку. Были рассмотрены следующие варианты:

AForge.NET и OpenCV в оболочке для .Net – Emgu CV. Выбор вариантов обусловлен необходимостью ограничиться свободно распространяемым программным обеспечением.

AForge.NET – фреймворк C# с открытым исходным кодом, предназначенный для разработчиков и исследователей в области компьютерного зрения и искусственного интеллекта. Решаются задачи обработки изображений, нейронных сетей, генетических алгоритмов, машинного обучения, робототехники [12]. Распространяется под лицензией LGPLv3.

OpenCV (Open Source Computer Vision Library) – это библиотека компьютерного зрения с открытым исходным кодом, включающая в себя алгоритмы компьютерного зрения, обработки изображений и численных алгоритмов общего назначения [8]. OpenCV имеет кросс-платформенную обертку для .Net – Emgu CV. Она позволяет вызывать функции OpenCV из совместимых с .NET языков, таких как C #, VB, VC ++, IronPython. Emgu CV может быть скомпилирована Visual Studio, Xamarin Studio, Unity и работает в Windows, Linux, Mac OS X, iOS, Android и Windows Phone [9]. Emgu CV распространяется под лицензией GPLv3 – лицензией на свободное программное обеспечение.

В ходе анализа [8], [9], [10], [12], [13] были выявлены достоинства и недостатки обоих библиотек. AForge.NET и OpenCV имеют широкий набор функция, хорошо проработанную документацию и открытый исходный код. AForge.NET более проста в использовании, в то время как OpenCV более популярна для решения задач в области компьютерного зрения. Однако при анализе производительности данных библиотек в [13] было выявлено следующее: AForge.NET значительно проигрывает OpenCV по времени выполнения одинаковых методов, что является большим недостатком.

Таким образом, для решения задач связанных с обработкой изображений была выбрана библиотека OpenCV.

Одной из основных целей OpenCV является предоставление простого в использовании интерфейса, который позволяет быстро строить сложные приложения, использующие компьютерное зрение. Библиотека OpenCV содержит более 500 функций, которые охватывают многие области компьютерного зрения [10].

OpenCV структурирована по пяти основным компонентам, четыре из которых показаны на рисунке 4.3. Компонент **CV** содержит основные алгоритмы обработки изображений и высокоуровневые алгоритмы компьютерного зрения. **ML –** библиотека машинного обучения, которая включает в себя средства статистической классификации и кластеризации. **HighGUI** содержит процедуры и функции ввода/вывода для хранения и загрузки видео и изображений. **CXCore** содержит основные структуры данных. Компонент **CvAux**, не представленный на рисунке 3.3, содержит устаревшие области и экспериментальные алгоритмы [10].

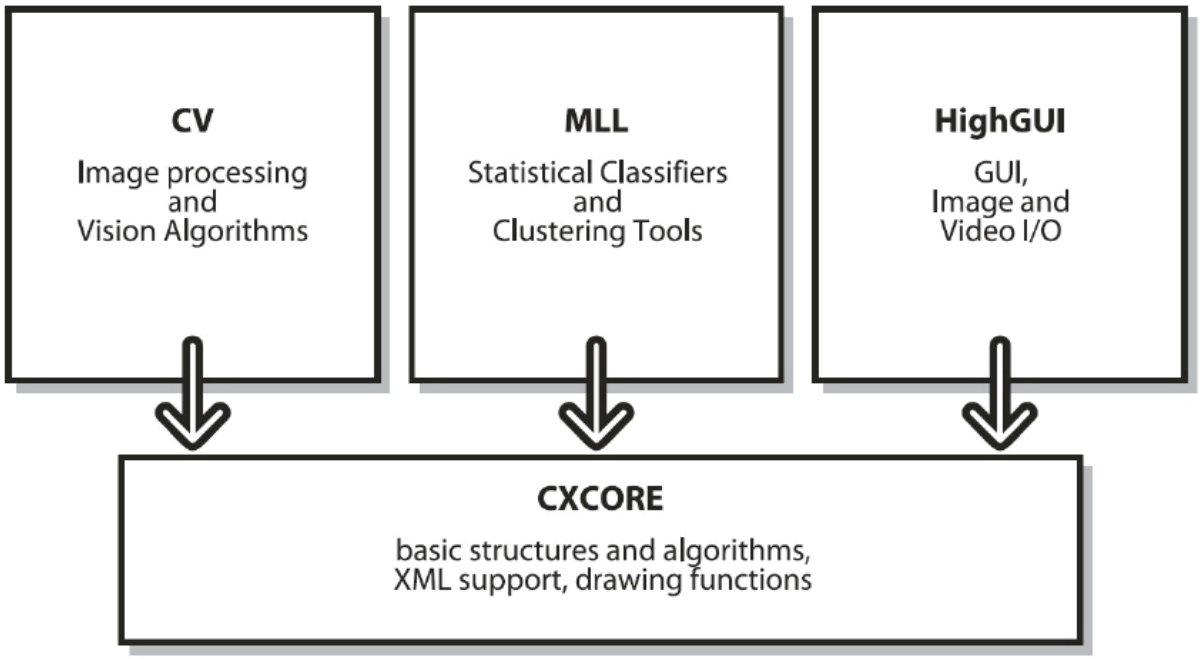

Рисунок 4.3 – Базовая структура OpenCV

Помимо OpenCV был выбран класс System.Drawing.Bitmap, обеспечивающий считывание и сохранение файлов различных графических форматов.

Итоговый набор инструментов разработки показан на рис. 4.4.

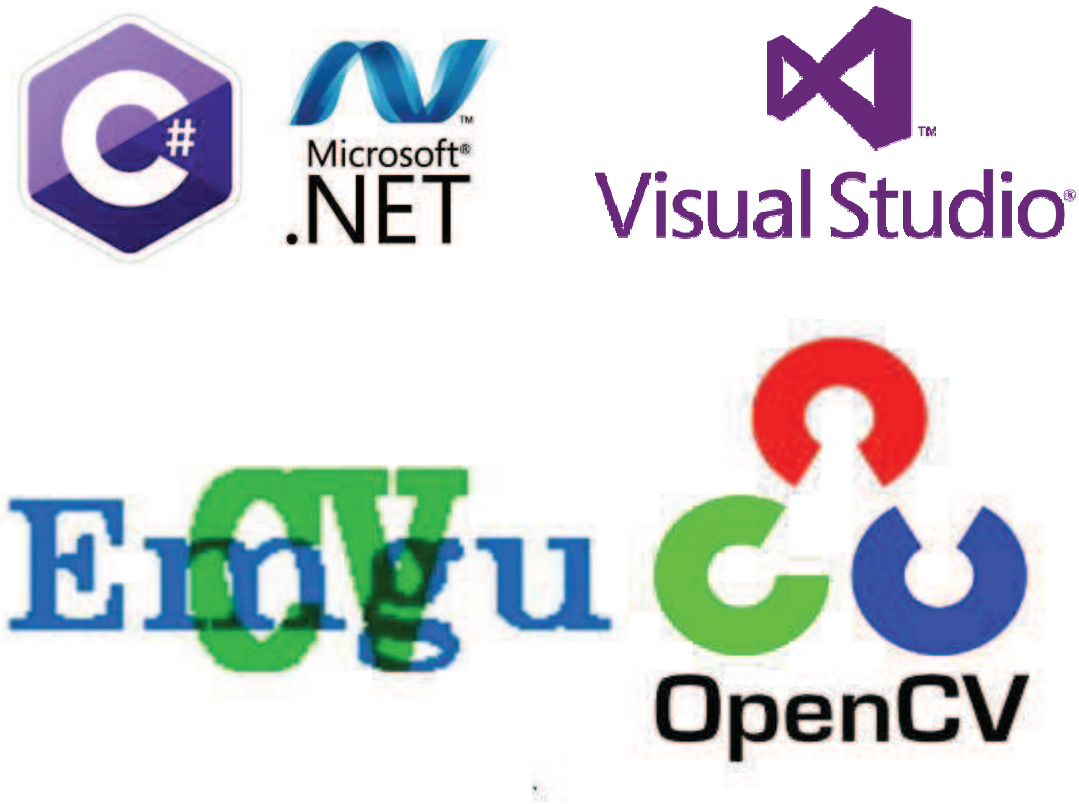

Рисунок 4.4 – Набор инструментов разработки

4.4. Программная реализация библиотеки обработки изображений

Библиотека обработки изображений предоставляет пользователю набор функционала для обработки отсканированных документов. Ниже описана реализация данного функционала.

4.4.1. Обрезка изображения

Обрезка изображения подразумевает выделение информативной части изображения и удаление лишнего фона. Данная задача сводится к поиску фрагментов на изображения с высокими перепадами яркости.

Алгоритм обрезки изображения представлен на рисунке 4.5.

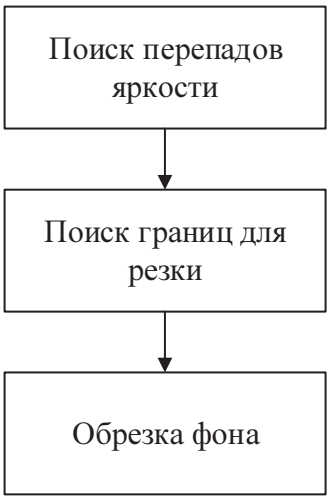

Рисунок 4.5 – Алгоритм обрезки изображения

Изображение разбивается на фрагменты, после чего подсчитывается количество перепадов яркости по горизонтали и вертикали для каждого фрагмента с сохранением этого значения и координат фрагмента. Фрагмент представляет собой квадрат, стороны которого равны одной пятидесятой ширины изображения. Между соседними пикселями фиксируется перепад яркости, если разница их интенсивности превышает пороговое значение, по умолчанию равное 25. Этот параметр при необходимости может быть настроен пользователем. Интенсивностью цвета для цветных изображения является среднее арифметическое трех цветовых компонентов.

Фрагмент кода поиска перепадов из метода RemoveBackground класса ImageConverter показан в листинге 4.1.

Листинг 4.1 – Фрагмент кода поиска перепадов яркости по горизонтали и вертикали из метода RemoveBackground класса ImageConverter.

```
//Поиск координат фрагментов с высоким перепадом яркости
for (int y = 0; y < bitmapData.Height + regionSize; y += regionSize)
{
 for (int x = 0; x < bitmapData.Width + regionSize; x += regionSize)
 {
   var value = 0;
   //Горизонтальный перебор отдельного фрагмента
   for (int yy = y; (yy < y + regionSize) && (yy < bitmapData.Height); yy++)
      {
        byte pixel = GetGrayPixel(sourceBytes, bitmapData.Width, x, yy);
```

```
            for (int xx = x; (xx < x + regionSize) && (xx < bitmapData.Width);
xx++)
              {
                byte nextPixel = GetGrayPixel(sourceBytes, bitmapData.Width,
xx, yy);
                if (Math.Abs(pixel - nextPixel) > sensitivity) value++;
                pixel = nextPixel;
                }
          }

       //Вертикальный перебор отдельного фрагмента
       for (int xx = x; (xx < x + regionSize) && (xx < bitmapData.Width); xx++)
       {
             byte pixel = GetGrayPixel(sourceBytes, bitmapData.Width, xx, y);

           for (int yy = y; (yy < y + regionSize) && (yy < bitmapData.Height);
yy++)
              {
                 byte nextPixel = GetGrayPixel(sourceBytes, bitmapData.Width,
xx, yy);
                 if (Math.Abs(pixel - nextPixel) > sensitivity) value++;
                 pixel = nextPixel;
               }
        }
        //Фиксируем значение количества перепадов яркости для фрагмента
        pointList.Add(new Point() { V = value, X = x, Y = y });
        maxV = Math.Max(maxV, value);
      }
   }
```

Затем необходимо найти границы резки – области с наибольшим перепадом яркости, так как фон от сканера предполагает однородность и, как следствие, малое количество перепадов яркости, а информативная область наоборот включает много перепадов яркости. Фрагмент кода поиска границ для резки из метода RemoveBackground класса ImageConverter показан в листинге 4.2.

Листинг 4.2 – Фрагмент кода с поиск границ для резки из метода RemoveBackground класса ImageConverter.

```
foreach (Point point in pointList)
{
 var v = (byte)(point.V * vFactor);
 if (v > treshold)
   {
     x1 = Math.Min(x1, point.X);
     y1 = Math.Min(y1, point.Y);

     x2 = Math.Max(x2, point.X + regionSize);
     y2 = Math.Max(y2, point.Y + regionSize);
     }
 }
```

По найденным координатам формируются границы, и изображение обрезается. Используется метод CropImage класса ImageConverter. Фрагмент кода обрезки изображения показан в листинге 4.3.

Листинг 4.3 – Фрагмент кода с обрезка изображения из метода CropImage класса ImageConverter.

```
        private Bitmap CropImage(Bitmap source, Rectangle section)
            {
                Bitmap bmp = new Bitmap(section.Width, section.Height);
                bmp.SetResolution(source.HorizontalResolution,
source.VerticalResolution);
                Graphics g = Graphics.FromImage(bmp);
                g.DrawImage(source, 0, 0, section, GraphicsUnit.Pixel);

                return bmp;
            }
```

4.4.2. Коррекция угла наклона по основным линиям

Коррекция угла наклона подразумевает приведение изображения к вертикальной или горизонтальной ориентации. Для этого необходимо выделить основные линии и определить их угол наклона.

Нахождение угла наклона изображения по основным линиям можно разбить на следующие этапы:

1. Предварительная обработка;
2. Поиск основных линий и расчет угла наклона.

Алгоритм коррекции угла наклона по основным линиям представлен на рисунке 4.6.

Рисунок 4.6 – Алгоритм коррекции угла наклона по основным линиям

Прежде чем осуществить поиск основных линий, их необходимо визуализировать. Для определения угла поворота используются горизонтальные линии. Они выделяются с помощью оператора Собеля, реализованного в OpenCV.

Оператор Собеля — дискретный дифференциальный оператор, вычисляющий приближённое значение градиента яркости изображения [14]. Он вычисляет градиент яркости в каждой точке. Так находится направление наибольшего увеличения яркости и величина её изменения в этом направлении. Результат показывает, насколько «резко» или «плавно» меняется яркость изображения в каждой точке, а значит, вероятность нахождения точки на грани, а также ориентацию границы [15].

Фрагмент кода с предварительной обработкой изображения из метода AngelCorrect класса ImageConverter показан в листинге 4.4.

Листинг 4.4 – Фрагмент кода с предварительной обработкой изображения из метода AngelCorrect класса ImageConverter.

```
//Преобразование изображения в полутоновое
UMat uimage = new UMat();
CvInvoke.CvtColor(img, uimage, ColorConversion.Bgr2Gray);
```

```
    //Сглаживание
    UMat pyrDown = new UMat();
    CvInvoke.PyrDown(uimage, pyrDown);
    CvInvoke.PyrUp(pyrDown, uimage);

    Image<Gray, byte> sobel = new Image<Gray, byte>(img.Size);
    //Применение оператора Собеля.
    CvInvoke.Sobel(uimage, sobel, DepthType.Cv8U, 0, 1, 3);
```

Поиск линий осуществляется с помощью метода HoughLinesP, реализованного в OpenCV.

Для вычисления угла наклона формируется среднее значение точек полученных линий и достраивается угол с помощью горизонтальной линии. Для получения значения угла воспользуемся геометрической формулой для нахождения угла между катетом и гипотенузой в прямоугольном треугольнике. Корректируется наклон с помощью метода Rotate, реализованного в OpenCV.

Фрагмент кода с поиском основных линий и расчетом угла наклона из метода AngelCorrect класса ImageConverter показан в листинге 4.5.

Листинг 4.5 – Фрагмент кода с поиском основных линий и расчетом угла наклона из метода AngelCorrect класса ImageConverter.

```
      //Поиск линий
   LineSegment2D[] lines = CvInvoke.HoughLinesP(sobel, 1, Math.PI / 180.0,
20, 50);

       if (lines != null && lines.Length > 0)
          {
            double angle = 0;
            LineSegment2D avr = new LineSegment2D();

           //Формирование среднего значения линий
            foreach (LineSegment2D seg in lines)
            {
               avr.P1 = new System.Drawing.Point(avr.P1.X + seg.P1.X,
avr.P1.Y + seg.P1.Y);
               avr.P2 = new System.Drawing.Point(avr.P2.X + seg.P2.X,
avr.P2.Y + seg.P2.Y);
            }

            avr.P1 = new System.Drawing.Point(avr.P1.X / lines.Length,
avr.P1.Y / lines.Length);
            avr.P2 = new System.Drawing.Point(avr.P2.X / lines.Length,
avr.P2.Y / lines.Length);

                    LineSegment2D horizontal = new LineSegment2D(avr.P1, new
System.Drawing.Point(avr.P2.X, avr.P1.Y));
```

```
            //Вычисление угла для поворота
             double c = horizontal.P2.X - horizontal.P1.X;
             double a = Math.Abs(horizontal.P2.Y - avr.P2.Y);
             double b = Math.Sqrt(c * c + a * a);
             angle = (a / b * (180 / Math.PI))*(horizontal.P2.Y>avr.P2.Y ? 1
: -1);

             Bgr color00 = img[0, 0];
             Bgr color01 = img[0, img.Cols - 1];
             Bgr color10 = img[img.Rows - 1, 0];
             Bgr color11 = img[img.Rows - 1, img.Cols - 1];

             Bgr fcolor = new Bgr();
             fcolor.Blue = (color00.Blue + color01.Blue + color10.Blue +
color11.Blue) / 4;
             fcolor.Green = (color00.Green + color01.Green + color10.Green +
color11.Green) / 4;
             fcolor.Red = (color00.Red + color01.Red + color10.Red +
color11.Red)/4;

           //Поворот изображения
             img = img.Rotate(angle, fcolor);
         }

      return img.ToBitmap();
   }
```

Недостатком данного метода является то, что поиск угла наклона осуществляется только по основным линиям, не учитывая положения обрабатываемого документа. Если документ на изображении расположен в неправильной ориентации, коррекция угла наклона будет произведена неправильно и отсканированный документ не будет приведен к своему стандартному виду. Поэтому необходимо применение метода, который учитывает его положение.

4.4.3. Коррекция угла наклона в соответствии с шаблоном

Коррекция угла наклона в соответствии с шаблоном сводится к поиску шаблона на изображении и определения положения, в котором корреляция интенсивности точек шаблона и изображения максимальна.

Алгоритм коррекции угла наклона в соответствии с шаблоном представлен на рисунке 4.7.

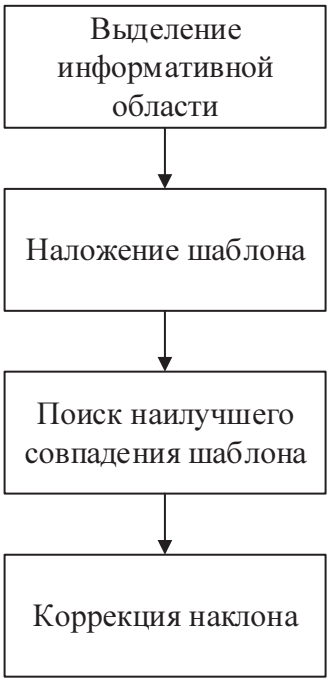

Рисунок 4.7 – Алгоритм коррекции угла наклона в соответствии с шаблоном

Шаблон для определения положения и типа документа представляет собой фрагмент документа, являющийся характерным признаком данного документа. Шаблоны хранятся в списке объектов класса Templates. Пример шаблона для обработки паспорта РФ представлен на рисунке 4.8.

Рисунок 4.8 – Шаблон для обработки паспорта РФ

В библиотеке OpenCV реализована функция MatchTemplate, сравнивающая шаблон с исходным изображением при помощи «скольжения» шаблоном над исходным изображением, используя метод корреляции. Для достижения инвариантности к вращению шаблона необходимо поворачивать его на некоторый угол и заново производить подсчет корреляции.

Согласно [10] в MatchTemplate подсчет результирующего значения корреляции производится по формуле (1), нормирование этого значения выполняется по формуле (2):

$$R(x,y) = \sum_{x',y'} [T(x',y') * I(x+x',y+y')]^2, \qquad (1)$$

$$R_{normed}(x,y) = \frac{R(x,y)}{Z(x,y)}, \qquad (2)$$

$$Z(x,y) = \sqrt{\sum_{x',y'} T(x',y')^2 * \sum_{x',y'} I(x+x',y+y')^2}, \qquad (3)$$

Где R – результат, T – шаблон, I – исходное изображение, Z – коэффициент нормализации.

Чтобы определить, находится ли шаблон на исходном изображении, необходимо значение наилучшего совпадения с пороговым. При этом для метода, основанного на корреляции, идеальное совпадение будет равно 1, а отсутствие совпадения – 0. Поэтому при вращении шаблона угол поворота определяется как наибольшее из всех полученных значений.

Фрагмент кода с поиском наилучшего совпадения шаблона из метода SearchingTemplates класса TemplatesCompare показан в листинге 4.6.

Листинг 4.6 – Фрагмент кода с поиском наилучшего совпадения шаблона из метода SearchingTemplates класса TemplatesCompare.

```
        CvInvoke.MatchTemplate(uimage, utemp, resultimg,
TemplateMatchingType.CcorrNormed);
        double minval = new double();
        double maxval = new double();
        System.Drawing.Point minloc = new System.Drawing.Point();
        System.Drawing.Point maxloc = new System.Drawing.Point();

        CvInvoke.MinMaxLoc(resultimg, ref minval, ref maxval, ref minloc, ref
maxloc);

        saveresult.Add(new ListforSearch(minval, maxval, minloc, maxloc,
(double)i));

        Bgr color00 = template[0, 0];
        Bgr color01 = template[0, template.Cols - 1];
        Bgr color10 = template[template.Rows - 1, 0];
        Bgr color11 = template[template.Rows - 1, template.Cols - 1];

        Bgr fcolor = new Bgr();
```

```
        fcolor.Blue = (color00.Blue + color01.Blue + color10.Blue +
color11.Blue) / 4;
        fcolor.Green = (color00.Green + color01.Green + color10.Green +
color11.Green) / 4;
        fcolor.Red = (color00.Red + color01.Red + color10.Red +
color11.Red)/4;

        template = template.Rotate(angelst, fcolor);
```

4.4.4. Определение формата изображения

Для определения формата изображения используется свойство Bitmap изображений RawFormat. Фрагмент кода проверки формата изображения из класса ImageConverter показан в листинге 4.7.

Листинг 4.7 – Фрагмент кода с проверкой формата изображения из класса ImageConverter.

```
    public Formats CheckImageFormat(byte[] sourcebyte)
    {
        Bitmap _image = ByteArrayToBitmap(sourcebyte);
        Formats fr = Formats.Unknown;
        if (System.Drawing.Imaging.ImageFormat.Jpeg.Equals(_image.RawFormat))
        {
          fr = Formats.JPEG;
        }
        else if
(System.Drawing.Imaging.ImageFormat.Tiff.Equals(_image.RawFormat))
        {
          fr = Formats.TIFF;
        }
        else if
(System.Drawing.Imaging.ImageFormat.Bmp.Equals(_image.RawFormat))
        {
          fr = Formats.BMP;
        }
        else if
(System.Drawing.Imaging.ImageFormat.Png.Equals(_image.RawFormat))
        {
          fr = Formats.PNG;
        }
        else if
(System.Drawing.Imaging.ImageFormat.Gif.Equals(_image.RawFormat))
        {
          fr = Formats.GIF;
        }

        return fr;
    }
```

4.4.5. Сохранение изображения с сжатием разрешения

Для сжатия разрешения изображения до максимально установленного размера используется функция Resize из OpenCV. Фрагмент кода с сжатием изображения из класса ImageConverter показан в листинге 4.8.

Листинг 4.8 – Фрагмент кода с сжатием изображения из класса ImageConverter.

```
public byte[] SaveWithCompression(byte[] imagearray, int maxresolution)
        {

            Bitmap sourceImage = ByteArrayToBitmap(imagearray);
            Image<Bgr, Byte> img = new Image<Bgr, byte>(sourceImage);
            int res;

            if (img.Width > img.Height) res = img.Width;
            else res = img.Height;

            if ((maxresolution > 0)&&(maxresolution<res))
            {
                double scale = maxresolution / res;
                img = img.Resize(scale, Inter.Cubic);
            }
            return BitmapToByteArray(img.ToBitmap(), sourceImage);
        }
```

5. РЕЗУЛЬТАТЫ РАЗРАБОТКИ

Результатом разработки является библиотека в формате dll, предоставляющая пользователю набор инструментов для обработки отсканированных документов. По требованию заказчика инструменты оформлены в виде методов.

Пользователю предоставляются следующие методы:

- RemoveBackground для удаления фона от сканера;
- AngelCorrect для коррекции углов наклона по основным линиям;
- AngelCorrectDocType для коррекции угла наклона в соответствии с шаблоном;
- CheckImageFormat для определения формата изображения;
- SaveWithCompression для сохранения изображения с компрессией.

5.1. Вызов метода RemoveBackground

Метод RemoveBackground отвечает за удаление фона от сканера. Результатом работы данного метода является обработанное изображение, содержащие только информативную область документа.

Пример работы метода RemoveBackground представлен на рисунке 5.1.

Рисунок 5.1 – Пример работы метода RemoveBackground (слева – оригинал, справа – обработанное изображение)

5.2. Вызов метода AngelCorrect

Метод AngelCorrect отвечает за коррекцию угла наклона по основным линиям. Результатом работы данного метода является изображение, приведенное к строго вертикальной или горизонтальной ориентации.

Примеры работы метода AngelCorrect представлены на рисунках 5.2 и 5.3. На рисунке 5.3 документ имеет неправильную ориентацию, а метод AngelCorrect, основываясь только на основных линиях, не способен корректно обработать изображение.

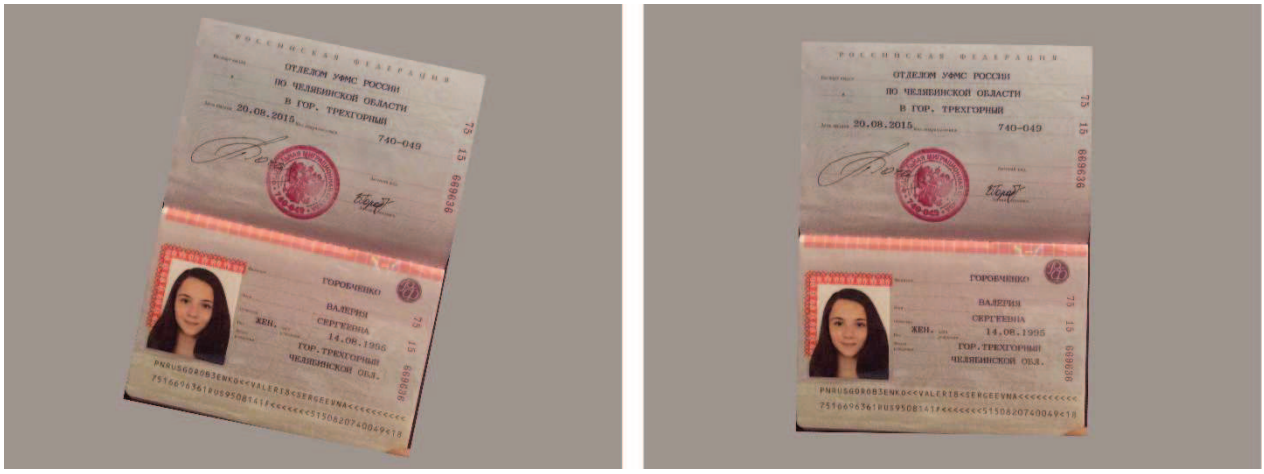

Рисунок 5.2 – Пример работы метода AngelCorrect (слева – оригинал, справа – обработанное изображение)

Рисунок 5.3 – Пример работы метода AngelCorrect (слева – оригинал, справа – обработанное изображение)

5.3. Вызов метода AngelCorrectDocType

Метод AngelCorrectDocType отвечает за коррекцию угла наклона в соответствии с шаблоном. Результатом работы данного метода является изображение, приведенное к своему стандартному виду.

Примеры работы метода AngelCorrectDocType представлены на рисунках 5.4 и 5.5. На рисунок 5.4 документ имеет правильную ориентацию, найденную благодаря методу AngelCorrectDocType.

Рисунок 5.4 – Пример работы метода AngelCorrectDocType (слева – шаблон для поиска, справа – обработанное изображение с найденным шаблоном)

Рисунок 5.5 – Пример работы метода AngelCorrectDocType (слева – шаблон для поиска, справа – обработанное изображение с найденным шаблоном)

5.4. Вызов метода CheckImageFormat

Метод CheckImageFormat отвечает за определение формата изображения по его двоичному коду. Результатом работы данного метода является формат изображения.

Пример работы метода CheckImageFormat представлен на рисунке 5.6.

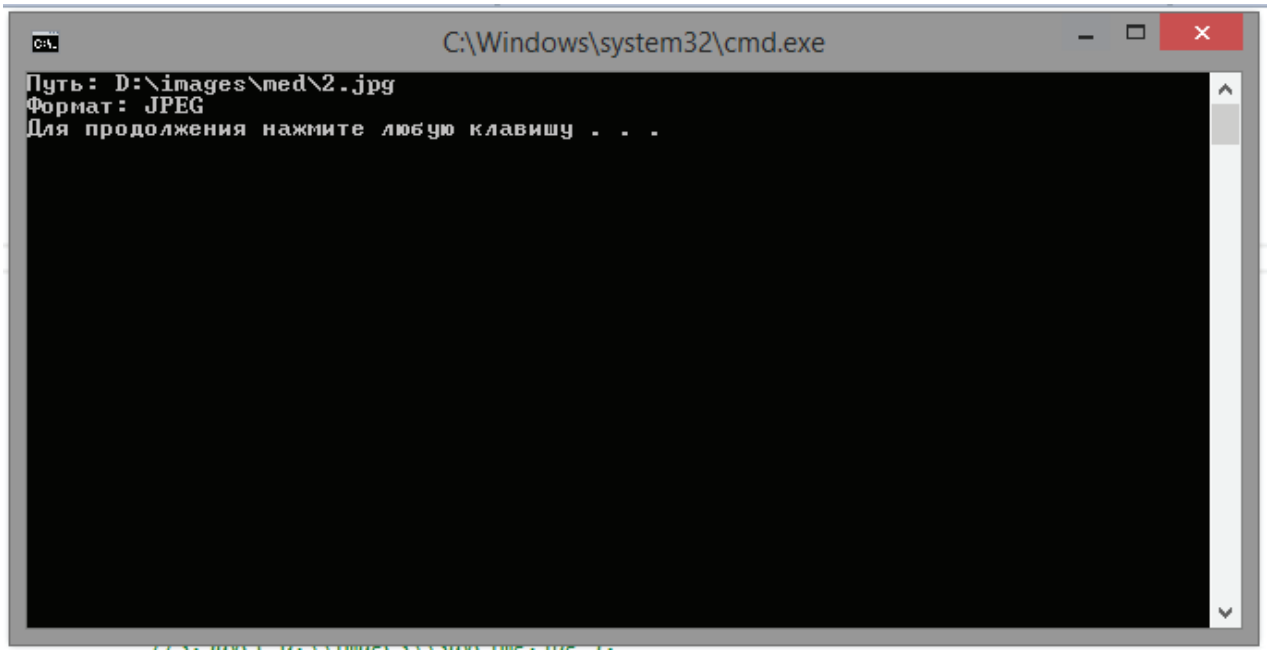

Рисунок 5.6 – Пример работы метода CheckImageFormat

5.5. Вызов метода SaveWithCompression

Метод SaveWithCompression отвечает за сжатие разрешения исходного изображения до максимально установленного. Результатом работы данного метода является изображение уменьшенного формата.

Пример работы метода SaveWithCompression представлен на рисунке 5.7.

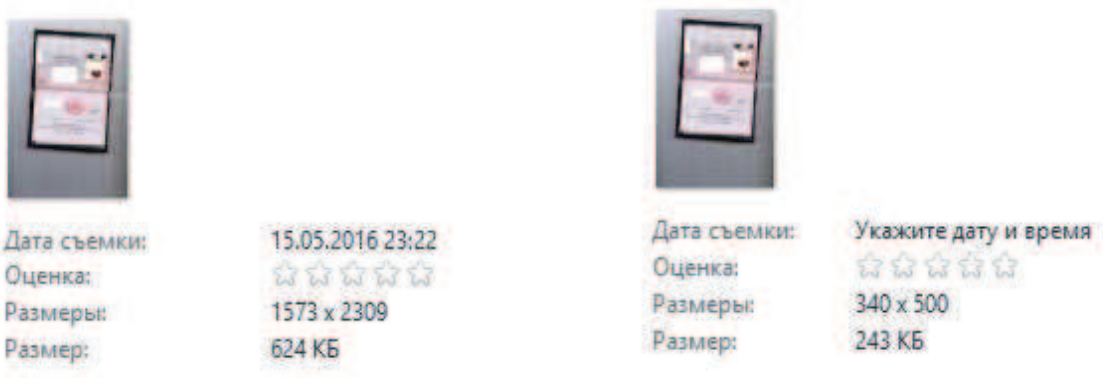

Рисунок 5.7 – Пример работы метода CheckImageFormat

ЗАКЛЮЧЕНИЕ

В результате проведенной работы были выявлены типовые дефекты документов, вводимых в систему Универис, проанализированы существующие решения, позволяющие обрабатывать изображения документов.

На основе полученных данных определен необходимый функционал для приведения изображений к стандартному виду:

- функция определения типа документа;
- функции определения положения документа;
- функции сохранения файла с сжатием разрешения изображения;
- функции сохранения файла в исходном разрешении;
- функции удаления фона от сканера;
- функции коррекции угла наклона документа;
- функции определения формата изображения.

Исходя из функциональных требований к создаваемой библиотеке были спроектированы диаграмма классов и диаграмма вариантов использования, выбраны инструменты разработки.

В итоге была разработана библиотека в формате dll, предоставляющая пользователю набор инструментов для обработки отсканированных документов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Четыре в одном: обзор нового ABBYY FineReader 14 / Программное обеспечение [Электронный ресурс]. – Электрон. текстовые дан. – Режим доступа: https://3dnews.ru/947258

2 Программа для распознавания текста ABBYY FineReader [Электронный ресурс]. – Электрон. текстовые дан. – Режим доступа: https://www.abbyy.com/ru-ru/finereader/

3 ABBYY PassportReader SDK [Электронный ресурс]. – Электрон. текстовые дан. – Режим доступа: https://www.abbyy.com/ru-ru/passportreader-sdk/

4 ScanTailor [Электронный ресурс]. – Электрон. текстовые дан. – Режим доступа: http://scantailor.org/

5 ScanKromsator — Википедия [Электронный ресурс]. – Электрон. текстовые дан. – Режим доступа: https://ru.wikipedia.org/wiki/ScanKromsator

6 Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, Н. Якобсон. – 2-е изд. – М.: ДМК Пресс, 2006. – 496с.

7 Microsoft Visual Studio — Википедия [Электронный ресурс]. – Электрон. текстовые дан. – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio

8 OpenCV — Википедия [Электронный ресурс]. – Электрон. текстовые дан. – Режим доступа: https://ru.wikipedia.org/wiki/OpenCV

9 Emgu CV: OpenCV in .NET (C#, VB, C++ and more) [Электронный ресурс]. – Электрон. текстовые дан. – Режим доступа: http://www.emgu.com/wiki/index.php/Main_Page

10 Bradski, G. Learning OpenCV / G. Bradski, A. Kaehler. – CA: O'Reilly Media, 2008. – 556p.

11 C Sharp — Википедия [Электронный ресурс]. – Электрон. текстовые дан. – Режим доступа: https://ru.wikipedia.org/wiki/C_Sharp

12 AForge.NET :: Framework [Электронный ресурс]. – Электрон. текстовые дан. – Режим доступа: http://www.aforgenet.com/framework/

13 Comparing image-processing libraries – Eye Power [Электронный ресурс]. – Электрон. текстовые дан. – Режим доступа: https://eyepowr.wordpress.com/2014/07/11/comparing-image-processing-libraries/

14 Оператор Собеля — Википедия [Электронный ресурс]. – Электрон. текстовые дан. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_%D0%A1%D0%BE%D0%B1%D0%B5%D0%BB%D1%8F

15 Оператор Собеля | Ivan Andreev's Blog [Электронный ресурс]. – Электрон. текстовые дан. – Режим доступа: https://iandreev.wordpress.com/2010/06/22/sobel/