СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире, в связи со стремительным развитием технологий и ростом их объемов, возникает необходимость в своевременном извлечении информации, содержащейся на цифровых изображениях, с целью использования этой информации в различных областях человеческой деятельности.

Одной из этих областей является производство, в котором после использования листового материала остаются деловые остатки.

Деловыми остатками (ДО) называют остатки материала от раскроя, которые складируют и, в дальнейшем, используют для производства деталей. Их рассматривают, как листы меньшего размера.

Актуальность. Вопрос использования деловых остатков актуален в настоящее время, поскольку его использование решает ряд задач производства по хранению, утилизации и повторном использовании остатков. Для повторного использования ДО, их необходимо измерить. Замеры вручную могут занять много времени, поэтому сфотографировав остатки, и взяв всю необходимую информацию о размерах из цифровой фотографии, значительно сократит время работы на производстве.

В научной и производственной сфере достаточно актуальна задача нахождения и распознавания объектов на изображении. В основном рассмотрены методы нахождения текстовой информации [1] и биометрических персональных данных [2-4]. Авторы многократно рассматривают алгоритмы распознавания лиц [2, 4] и нахождения контурной границы [5-6].

Объектом исследования является деловые остатки на производстве.

Предметом исследования является процесс определения геометрии и размеров ДО.

Цель работы заключается в разработке программного обеспечение получения размеров делового остатка нестандартной формы, по цифровой фотографии.

Задачи. Для реализации поставленной цели, необходимо выполнить следующие задачи:

1. Проанализировать анализ предметной области;
2. Разработать математическую модель;
3. Разработать интерфейс приложения;
4. Описать основные функции системы
5. Экономически обосновать.

Реализовать систему средствами языка программирования python 3.8 32 bit, на основе библиотек программных модулей.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Анализ деятельности персонала для получения размеров делового остатка нестандартной формы.

На предприятиях ДО складируется или при невозможности, его повторного использования, утилизируется. При использовании минимизируются отходы производства, снижая затраты производства на материал.

При хорошем освящении и с подходящим фоном, работник делает фотографию ДО.

В дальнейшем фотография загружается в ПО, и получает данные с площадью и БД, в которой содержатся координаты контура делового остатка, для дальнейшего использования на предприятии.

Исходя из этого, действующим лицом (актером) данной предметной области является персонал предприятия, наглядное изображение деятельности получения размеров делового остатка представлено на рисунке 1.

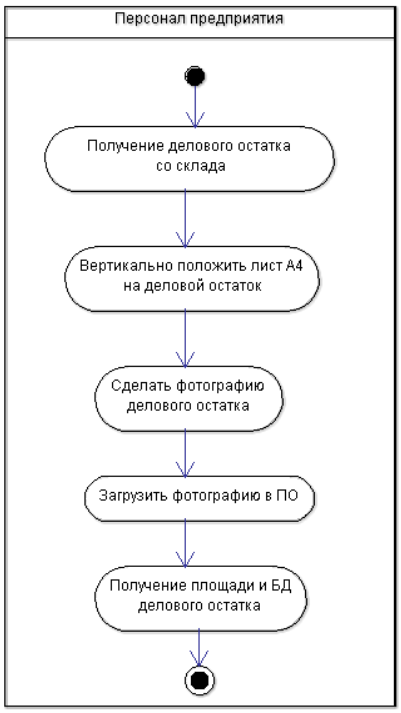


Рисунок 1 - Диаграмма деятельности предметной области

Для проектирования и реализации ПО получения контура с реальными размерами и площадью делового остатка, создана модель AS-IS.

Реализуя модель, использован метод SADT, предложенный Д. Россом. На основе этого метода разработана методология IDEF0.

Результат применения метода SADT - модель, которая состоит из диаграмм IDEF0, фрагментов текстов и глоссария со ссылками друг на друга. Все функции и интерфейсы представляются диаграммами в виде, соответственно, блоков и дуг. Место соединения дуги с блоком определяет тип интерфейса. Управляющая информация входит в блок сверху, в то время как информация, которая подвергается обработке (исходные данные), указывается с левой стороны блока, а результаты работы функции (выход, результат) - с правой стороны. Механизм, осуществляющий операцию (человек или автоматизированная система), задается дугой, входящей в блок снизу.

Создана контекстная диаграмма предметной области, согласно таблице 1.

Таблица 1. Контекстная диаграмма

Наименование стрелки	**Тип**
Деловой остаток	Вход
Персонал предприятия	Механизм
Нормативно-правовая база	Управление
Площадь делового остатка	Выход
Контур делового остатка	Выход

В результате получена контекстная диаграмма, представленная на рисунке 2.

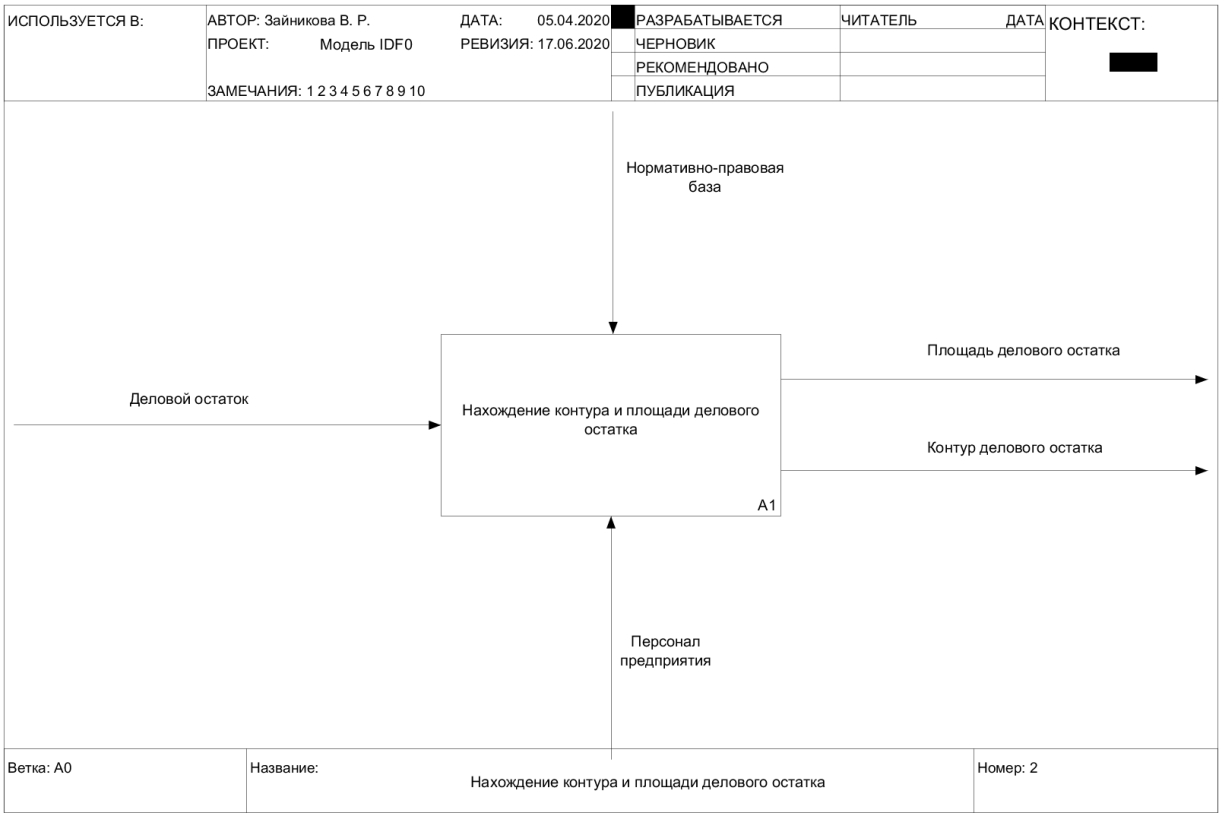


Рисунок 2- Контекстная диаграмма

Персонал предприятия осуществляет все операции внутри системы. Входными данными является деловой остаток. Управляющей информацией является нормативно правовая база. Результатом является площадь ДО и контур, содержащий его координаты.

Для уточнения механизма работы системы, созданы диаграммы декомпозиции, представленные на рисунках3 и 4.

Диаграмма, представленная на рисунке 4, включает в себя все необходимые функции для получения конечного результата. Они представлены в виде блоков, стрелками указана их связь и последовательность действий.

Функции системы:

1) Сфотогрофировать ДО;

2) Загрузка фотографии с изображением ДО;

3) Исправление перспективы для получения изображениявида «сверху - вниз»;

4) Автоматическое выделение контура с возможностью его редактирования;

5) Расчет площади выделенного контура;

6) Ввод данных контура в базу данных.

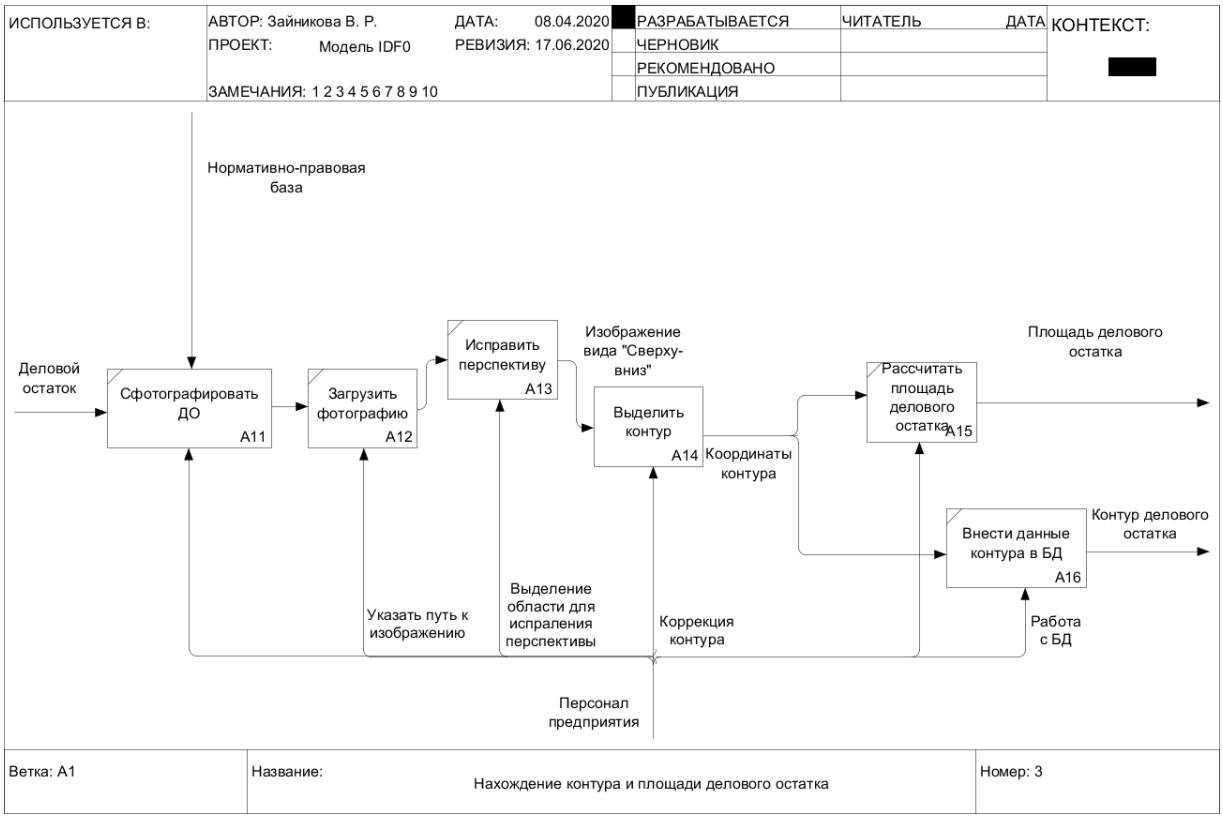


Рисунок 3- Контекстная диаграмма

Диаграмма, представленная на рисунке 6, является дочерней для диаграммы, представленной на рисунке 5. Она включает в себя следующую последовательность действий блока «Выделить контур»:

1) Найти контур листа А4;

2) Изменить размеры изображения относительно листа А4;

3) Выделение контура ДО.

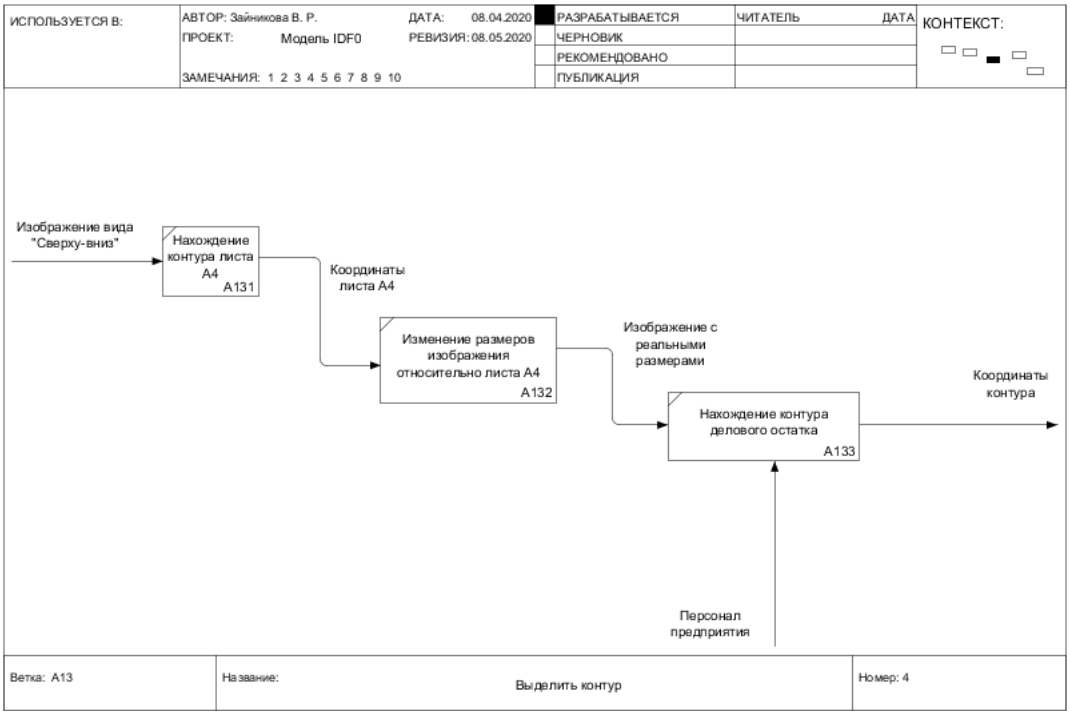


Рисунок 4- Контекстная диаграмма блока «Выделить контур»

Иерархия функциональных блоков представлена на рисунке 5.

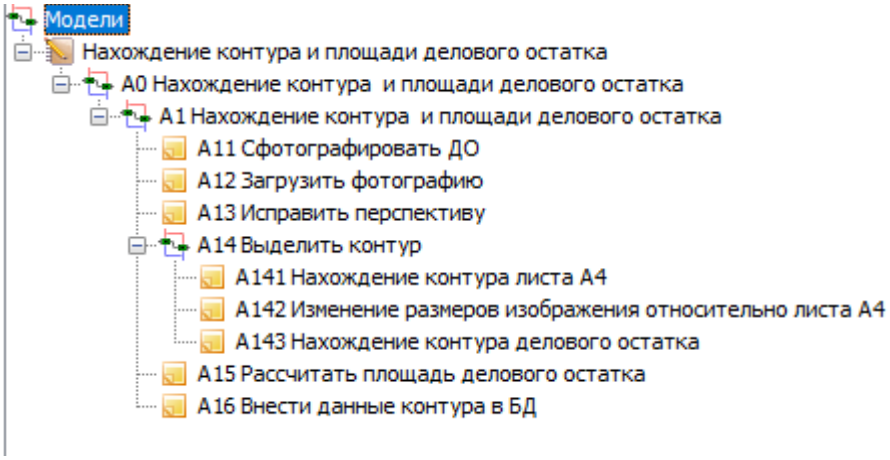


Рисунок 5- Иерархия функциональных блоков модели

Таким образом, произведен анализ предметной области. Выявлены основные критерии расположения ДО на цифровой фотографии. Обозначено действующее лицо, использующие программу, персонал предприятия. Обозначен функционал для проектирования и реализации ПО, получения контура с реальными размерами и площадью делового остатка методом функционального моделирования SADT (IDEF0). Обозначены входные и выходные данные.

1.2. Анализ существующих решений для получения размеров делового остатка нестандартной формы.

Подобная задача была рассмотрена в работе [8]. В ней рассмотрен алгоритм контурной сегментации и распознавания образцов объектов. В данном исследовании, над изображением, проведена предварительная обработка, далее выделены контуры для формирования словаря признаков, после чего идет сопоставление образцов с эталонами из базы данных. Данный метод лишь частично подходит для реализации системы, так как в нем, рассмотрено только нахождение контура, без его приведения к реальным размерам.

Задача обнаружение объекта включает в себя распознавание объекта, присутствующего в изображении, а также локализацию, чтобы вычислить ограничивающую рамку вокруг него. Эта проблема в области компьютерного зрения изучена в работах [9, 10, 11, 12]. В этих методах обычно используется информация о цветной камере. Во многих средах методы, основанные на сверточных нейронных сетях (CNNs), демонстрируют очень высокую точность обнаружения объектов. Тем не менее, эти методы уязвимы к ошибкам из-за качества цветных изображений, освещения и отсутствия цветового дифференцирования, для обнаружения объекта.

В исследовании [13] для нахождения объектов и их размеров на изображении, собираются мультимодальные данные с помощью пользовательского приложения Android. Так же, выполняется предварительная обработка данных для выполнения пространственно-временного выравнивания информации датчика о камере. Пространственное выравнивание требуется из-за разницы в положении камеры на мобильном устройстве. Временное выравнивание требуется из-за временного промежутка между сбором данных камеры. Таким образом, учитывается разница в положении устройства в разных случаях, когда собирается информация. После этого выравнивания получают информацию о глубине для некоторых пикселей цветных изображений камеры. Для получения информации о глубине для всех пикселей, выполняется

билинейная интерполяция и используется k-d-дерево, для быстрого поиска ближайшего пикселя, с информацией о глубине. Реализуется несколько методов расчета размеров интересующего объекта в метрах и показывается работоспособность в различных средах, как внутри помещения, так и снаружи.

Основными проблеммами данного метода являются: уязвимость к ошибкам из-за качества цветных изображений камеры, освещение и отсутствие цветового дифференцирования для обнаружения объекта. Так же, он имеет большую погрешность в расчетах, задокументировано, что датчики используемые в [13], демонстрируют погрешности равные 4 см. Даже при более высокой точности используя лидарные датчики погрешность равна 2 см.

Таким образом, рассмотрены существующие модели решения задачи, такие как нахождение контура и выделение области с обьектом, для дальнейщего определения его размеров. Описан их принцип действия и выявлены их основные недостатки.

1.3. Анализ инструментов и сред разработки для получения размеров делового остатка нестандартной формы

Для реализации ПО получения размеров ДО по цифровой фотографии, необходимо использовать инструметы библиотек компьютерного зрения.

Компьютерное зрение или машинное зрение представляет собой совокупность программно-технических средств, для обработки цифровых изображений и получения результатов, пригодных для практического использования.

В возможности библиотек машинного зрения входят решения практических задач по анализу содержимого фотографий, поиск и выделение текста, распознование обьектов и многое другое. Одной из библиотек решающей задачи продвинутой функциональности отностися Halcon, так же есть библиотека Libmv, рещающая задачи конкретной области, но одной из самых активно использующейся в различных сферах библиотекой является OpenCV.

Библиотека OpenCV реализована на кросплатформенном языке программирования C++ и располагает алгоритмами обработки изображения, а также встроенными алгоритмами с открытым кодом, позволяющие реализовать большую часть операций над изображениями. Библиотека может быть использована на платформах - Windows, Linux, Mac, Android, IOS. Так же имеется наличие интерфейсов для языков программирования: Python, Java, MATLAB, C ++

Несмотря на все достоинства библиотеки OpenCV, она имеет существенные недостатки, такие как, перегруженность второстепенными функциями, слабая документация, трудности в отладке программ, а так же сложность в изучении.

Таким образов, рассмотрена библиотека OpenCV, для реализации ПО получения размеров ДО по цифровой фотографии. Описаны ее основные преимущества и недостатки.

2. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

2.1. Математическая постановка задачи для получения размеров делового остатка нестандартной формы

Используя фотографию ДО нестандартной формы необходимо получить контур, содержавший в себе, n точек границ контура. Реализовать функцию получения площади делового остатка.

ДО на фотографии может находиться под любым углом в соответствии с рисуноком 6.

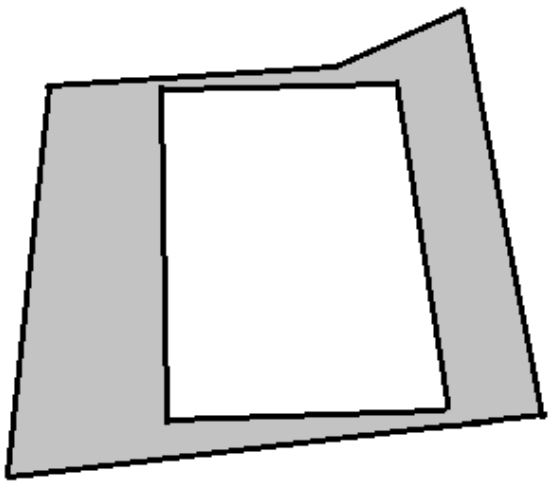


Рисунок 6 – Пример расположения ДО и листа А4

Для приведения изображения к реальным размерам, на ДО необходимо разместить лист формата А4, с установленным стандартом ISO 216. Это обусловлено тем обстоятельством, что размеры листа формата А4 известны и неизменны.

Главное условие - ДО, должен поместиться на фотографии. Соблюдение этого условия влияет на точность измерения. Располагать лист А4, необходимо, вертикально на поверхности ДО (рисунок 6).

С учетом проанализированной предметной области сформирован порядок действий, для реализации алгоритма нахождения контура ДО изображенного на рисуноке 7.

Рисунок 7 – Алгоритм нахождения контура ДО

Таким образом, необходимо реализовать функции для исправления перспективы, приведения изображения к реальным размерам и нахождение контура ДО.

2.2. Выбор инструментальных средств получения размеров делового остатка нестандартной формы

OpenCV поддерживает широкий спектр языков программирования и для реализации алгоритма, использован язык Python 3.8.

По сравнению с другими языками, Python работает медленнее, но его главной особенностью является то, что он может быть легко расширен с помощью C/C++. Эта особенность помогает писать коды в C/C++ и создавать для них оболочку Python, чтобы использовать их в качестве программных модулей. Это дает два преимущества: во-первых, код так же быстр, как исходный код C/C++ (так как это фактический код C++, работающий в фоновом режиме), а во-вторых, он очень легко кодируется на Python. OpenCV в Python, это оболочка вокруг оригинальной библиотеки C++.

Все структуры массива OpenCV преобразуются из изображения в массив NumPy. Это упрощает его интеграцию с другими библиотеками.

NumPy—это одна из основных Python-библиотек с поддержкой массивов. Изображение представляет собой стандартный массив NumPy, содержащий пиксели точек данных. Таким образом, при выполнении основных NumPy - операций можно изменять пиксельные значения изображения. Например, такие библиотеки, как SciPy и Matplotlib.

Таким образом, OpenCV-Python является подходящим инструментом для реализации задач компьютерного зрения. В особенности получения размеров делового остатка по цифровой фотографии.

2.3. Модель получения контура с рельными размерами делового остатка нестандартной формы

2.3.1. Исправление перспективы

Исправление перспективы изображения необходимо для получения вида «сверху - вниз» (см. рисунок 8) из цифровой фотографии. Для исправления перспективы использована библиотека алгоритмов обработки изображений, компьютерного зрения, а так же численных алгоритмов общего назначения OpenCV и библиотека NumPy.

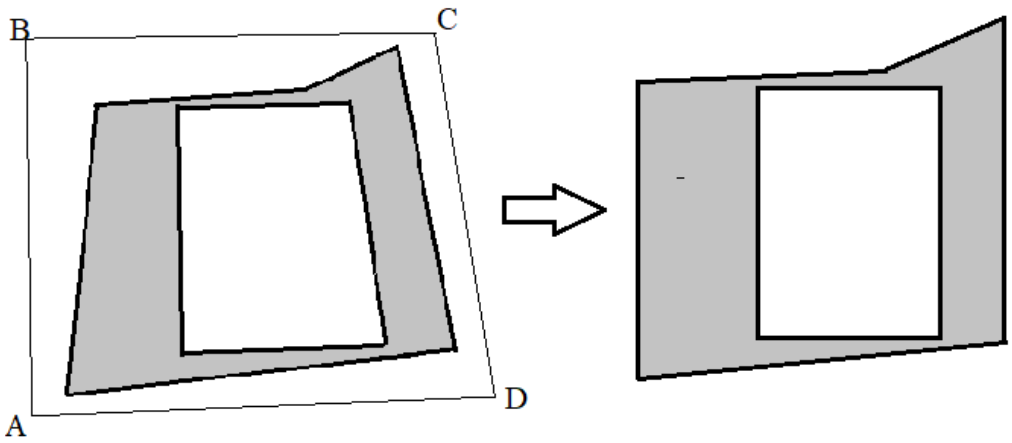


Рисунок 8 – Преобразование перспективы

Для исправления перспективы, ДО вписан в прямоугольник ABCD (см. рис. 3) и при помощи его вершин вычислена матрица преобразования перспективы. После нахождения матрицы применяете преобразование перспективы к изображению.

Используя функцию cv2.getPerspectiveTransform(src, dst) найдена матрица преобразования перспективы. Эта функция требует двух аргументов, список координат вершин описывающего четырехугольника src, и dst, который является списком преобразованных точек. То есть cv2.getPerspectiveTransform возвращает матрицу M, которая вычисляет 3×3 матрицу аффинного преобразования так, что:

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = map\_matrix \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \quad (1)$$

Где dst(i)=(x_i', y_i'), src(i) =(x_i, y_i), j=0, 1, 2, 3.

Матрица применятся в функции cv2.warpPerspective(src, M, dsize) → dst, результатом которой является желаемое изображение вида «сверху - вниз». Эта функция требует три аргумента, входное изображение src, матрица преобразования перспективы M и размеры выходного изображения dsize. Функция warpPerspective преобразует исходное изображение с помощью матрицы:

$$\mathrm{dst}(x, y) = \mathrm{src}\left(\frac{\mathrm{M}_{11}x + \mathrm{M}_{12}y + \mathrm{M}_{13}}{\mathrm{M}_{31}x + \mathrm{M}_{32}y + \mathrm{M}_{33}}, \frac{\mathrm{M}_{21}x + \mathrm{M}_{22}y + \mathrm{M}_{23}}{\mathrm{M}_{31}x + \mathrm{M}_{32}y + \mathrm{M}_{33}}\right) \quad (2)$$

Таким образом, применяя функции cv2.getPerspectiveTransform cv2.warpPerspective получено изображение с исправленной перспективой вида «сверху - вниз».

2.3.2. Приведение изображения к реальным размерам

Получив изображение с исправленной перспективой необходимо найти контур листа А4. Перед нахождением контуров необходимо провести предварительную обработку изображения. Задачей предварительной обработки является эффективное подавление шума при сохранении важных для последующего распознавания элементов изображения [14]. Использование возможностей обработки изображений значительно расширились за последние годы [15], охватывая такие области, как криминалистика и медицина [16].

Для подавления шума использован фильтр Гаусса [17] (см. рисунок 9). Для этого применим функцию cv2.GaussianBlur(src, ksize, sigmaX) → dst. Первый параметр src, это массив фотографии, далее ksize. размер ядра фильтра, чем больше значение, тем больше размытие (значения должны быть не четными). Далее идет sigmaX, стандартное отклонение по оси X . В нашем случае, размер ядра размытия (5, 5).

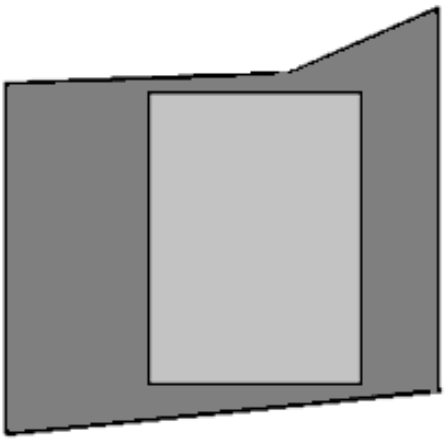


Рисунок 9 – Отфильтрованное изображение

Различное множество алгоритмов определения объектов рассматриваются в работе [18]:

1. Примитивы Хаара;
2. Adaptive Boosting;
3. Механизм AdaBoost;
4. Метод Виолы — Джонса;
5. Алгоритм Хафа;
6. Алгоритм сравнения с шаблоном;
7. Сегментация изображения.

Выделение контуров является одним из методов сегментации изображений [14, 19-21]. Под сегментацией принято понимать разбиение изображения на группы пикселей по близким значениям некоторого показателя [22]. Самые распространеные алгоритмы, основанны на выделении контурных границ с помощью вычисления первой производной (операторы Собеля и др. [23, 24]) или второй производной (оператор Лапласа [23]). В этом случае осуществляется поиск границ, которые в дальнейшем объединяются в контуры. В исследовании, приведённом в [25], показано, что методы пограничной обработкой уступают по качеству выделения контуров методу, предложенному Canny [17].

Детектор границ Canny включает следующую последовательность действий. После фильтрации помех на изображении гауссовым фильтром с заданным параметром сглаживания σ в каждой точке изображения вычисляется градиент, который характеризуется значением модуля

$$g(x, y) = \left[G_x^2 + G_y^2\right]^{\frac{1}{2}}, \quad (3)$$

и направлением

$$\alpha(x, y) = arctg\left[\frac{G_x}{G_y}\right]. \quad (4)$$

После получения величины градиента и направления, производится полный анализ изображения, чтобы удалить любые нежелательные пиксели, которые не могут составлять ребро. Результатом является двоичное изображение с "тонкими краями"(см. рисунок 10).

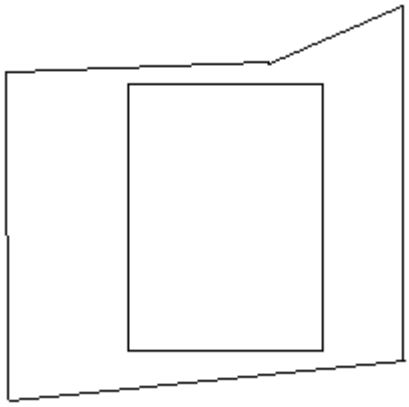


Рисунок 10 – Детектор границ Canny

Чтобы применить детектор границ Canny, необходимо использовать функцию cv2.Canny(image, threshold1, threshold2) → edges. Функция имеет 3 аргумента, входное изображение image, одноканальное изображение для обработки (градации серого), threshold1 — порог минимума, threshold2 — порог максимума.

Для перевода изображения в одноканальное используется функция cv2.cvtColor(image, cv2.COLOR_BGR2GRAY), она принимает два аргумента и

отдает итоговое изображение. Первый аргумент – исходное изображение, второй – направление трансформации цветового пространства.

Получив двоичное изображение найдены координаты всех контуров в изображении функцией cv2.findContours(image, mode, method) → contours, hierarchy. Эта функция требует три аргумента, image изображение, полученное детектором границ Канни. Mode режим группировки контуров. Использован режим CV_RETR_LIST для извлечения контуров без установления каких-либо иерархических связей. Method метод упаковки контуров. Использован метод CV_CHAIN_APPROX_SIMPLE, который склеивает все горизонтальные, вертикальные и диагональные контуры и оставляет только их конечные точки. Выходными данными функции являются contours и hierarchy.В contours каждый контур хранится в виде вектора точек, а hierarchy необязательный выходной вектор, содержащий информацию о топологии изображения.

Для проверки, является ли контур листом А4 или нет, выполнен цикл по каждому контуру. Для каждого из контуров вычислен периметр, используя функцию cv2.arcLength, далее аппроксимируем (сглаживаем) контур, используя cv2.approxPolyDP.

Причина, по которой используется аппроксимация контура, заключается в том, что он может не быть идеальным прямоугольником. Из-за зашумления и теней на фото вероятность того, что у листа будет ровно 4 вершины, невелика. Аппроксимируя контур, эта проблема решена.

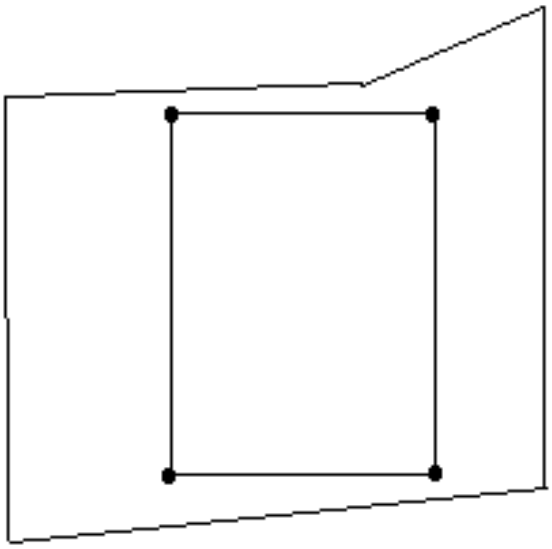


Рисунок 11 – Координаты листа А4

В нашем случае функции выглядят как:

peri = cv2.arcLength(contours, True)

approx = cv2.approxPolyDP(contours, 0.02 * peri, True)

Функция cv2.approxPolyDP принимает аргументы контур contours и 0.02 * peri параметр, задающий точность аппроксимации. Результат выполнения approx, вектор координат контура. Если количество координат равно четырем, то контур является четырехугольником (см. рисунок 11).

После нахождения четырехугольника, по его координатам, найдены длины его сторон вычислив евклидово расстояние между точками *i* и *j*.

$$d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\,. \qquad (5)$$

Рассчитав ширину и длину листа А4 на изображении, найдем, во сколько раз необходимо уменьшить или увеличить ширину и длину изображения, относительно реальных размеров листа А4, чтобы пиксельное значение ширины и длины изображения в миллиметрах, соответствовало реальным размерам. Вычислив новые значения ширины и длины изображения, с помощью библиотеки Image, изменим размеры изображения на новые. используя функцию Image.resize(size). Эта функция требует один аргумент size, запрошенный размер в пикселях, как 2-кортеж: (ширина, высота). Результатом выполнения функции является изображение заданного размера.

Для проверки корректно ли был найден лист А4 на изображении, повторно находим его координаты на новом изображении, с реальными размерами. Используя описанный выше порядок действий.

Далее находим площадь контура листа А4, используя функцию cv2.contourArea(contour). Она требует один аргумент contour, это входной вектор вершин контура, в нашем случае вершин листа А4. Если лист найден корректно, то площадь удовлетворяет условиям:

$$\begin{cases} S \geq 62338 \\ S \leq 62402 \end{cases} \quad (6)$$

Площадь листа А4 равна 62370 мм^2, согласно условию площадь может иметь отклонение на 32 мм^2.

$$p = \frac{100 \cdot 32}{62370} = 0{,}05\ \% \quad (7)$$

Таким образом, погрешность измерения не будет превышать ±0,05%. Данная погрешность по ГОСТу [26] соответствует измерению штангенциркулем. Получили изображение, которое соответствует реальным размерам, где 1 пиксель соответствует 1 мм.

2.3.3. Нахождение контура делового остатка

Для нахождения контура ДО используем функцию cv2.Canny, чтобы получить 8-битное одноканальное изображение.

Однако, на изображении, между вертикальными и горизонтальными полосками контура может оказаться пустое пространство. Чтобы его закрыть произведем ряд простых морфологических операций [27].

Начнем с создания прямоугольника между полосами с помощью функции cv2.getStructuringElement(shape, ksize) → kernel. Функция требует два аргумента shape и ksize. Shape форма элемента, в нашем случае четырехугольник MORPH_RECT. Размер структурирующего элемента ksize. Функция строит и

возвращает структурирующий элемент kernel, который передан в функцию morphologyEx().

Функция cv2.morphologyEx(src, op, kernel) → dst¶ производит морфологическую операцию, применив ядро к двоичному изображению, замазывая пространство между полосками. Функции необходимо три аргумента src, op и kernel. Src двоичное изображение, op тип морфологической операции, в нашем случае MORPH_CLOSE, для закрытия пустого пространства. Kernel структурирующий элемент.

$$\mathrm{dst}(x, y) = \mathrm{close}(\mathrm{str},\ \mathrm{element}) = \mathrm{erode}(\mathrm{dilate}(\mathrm{str},\ \mathrm{element})) \quad (8)$$

После морфологических операций над изображением находим контур делового остатка функцией cv2.findContours(). Используя режим группировки RETR_EXTERNAL для извлечения внешних контуров, и метод CV_CHAIN_APPROX_SIMPLE.

Чтобы получить контур делового остатка, отсортируем найденные контуры с помощью функции sorted(), которая возвращает новый отсортированный список. С помощью параметра key можно указывать, как именно выполнять сортировку. Флаг reverse позволяет управлять порядком сортировки. По умолчанию сортировка будет по возрастанию элементов.В нашем случае параметр key равен cv2.contourArea, выполняя сортировку по площади контура.

Отсортировав контуры сохраняем только самый большой из них, который содержит контур ДО (см. рисунок 12).

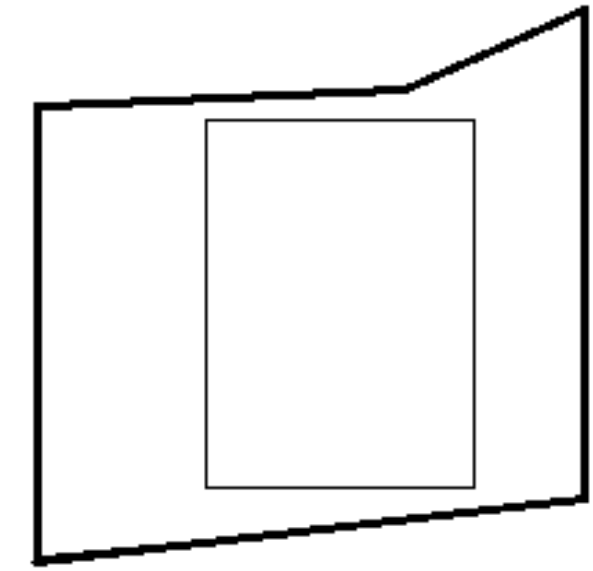


Рисунок 12 – Контур ДО

Таким образом, обработав изображение детектором границ Canny и морфологической операцией morphologyEx(), найдены контуры изображения. Отсортировав их и сохранив только самый большой, нашли контур ДО, который так же содержит в себе его координаты.

2.4. Результаты проведения эксперимента, применения модели нахождения ДО

В разработанном алгоритме исспользован метод обработки изображения для получения контура из работы [8]. Но он был улучшен, за счет исправления перспективы, выделяет более точный контур объекта.

В отличие от метода, рассмотренного в работе [12], точность нахождения ДО значительно выше, так как погрешность нахождения размеров разработанной модели равна 0,05%, а в методе она превышает 1 %. Главным недостатком является то, что в методе, не выделяется контур, объекты вписывают в прямоугольник, и выводят его размер, в результате получая только длину и ширину ДО. Результатом разработанного алгоритма, является контур ДО с его координатами.

Кроме того, обнаружение объектов на основе только цветных изображений, может обеспечить относительный размер в пикселях, который не имеет никакого отношения к реальным размерам изображения. Игрушечный автомобиль рядом с камерой, может казаться таким же большим, как настоящий автомобиль, который находится далеко от камеры.

Трудоемкость алгоритма и метода [8] различается в том, что в разработанной модели, объект на фотографии может быть изображен под любым углом, а в методе [8], для выделения более точного контура, нужно выполнить ряд действий, для получения изображения объекта вида «сверху-вниз» .

Скорость работы алгоритма не критично зависит от размера изображения (см. таблицу 2).

Таблица 2 – Время работы программы

Размер изображения	**Время**, с
434x422	0,098
612x582	0,106
1024x992	0,113
2000x1900	0,125
3456x4608	0,146

Таким образом, проанализирована работоспособность алгоритма. Выявлены основные плюсы алгоритма, в отличие от рассмотренных методов.

Вывод:

Разработана модель нахождения контура ДО нестандартной формы по цифровой фотографии с помощь. языка программирования Python, с использованием библиотеки компьютерного зрения OpenCV.

Функция исправления перспективы позволяет получить изображение вида «сверху-вниз». Контуры, полученные с этого преобразования, точнее передают форму ДО. Повышая точность для дальнейшего использования листового материала.

Используя метод сегментации, который играет важную роль в системах компьютерного зрения для решения задач, связанных с распознаванием и выделением объектов, были найдены все контуры в изображении. Используя апроксимацию для нахождения листа А4, найдены его коодинаты. Вычислив евклидово расстояние между координатами и сравнив с известными размерами листа А4, преобразовали изображение к точному размеру, где 1 пиксель в изображении, равен 1 мм.

Выполнив ряд морфологических операций, получен единый замкнутый контур с точными размерами, который содержит все координаты ДО.

3. ТЕХНОЛОГИЯ РЕАЛИЗАЦИИ

3.1. Общая концепция реализации программного обеспеченияполучения размеров делового остатка нестандартной формы

Программа реализована средствами языка программирования python для работы с изображением, использованы библиотеки программных модулей (см. рисунок 13). Из-за особенности библиотеки OpenCV [7], путь к проекту и фотографии ДО, не должен содержать русских слов.

```
import os, shutil
import sys
import cv2
import numpy as np
import math
from PIL import Image
from scipy.spatial import distance as dist
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
from matplotlib.artist import Artist
from matplotlib.patches import Polygon
from comtypes.client import CreateObject
from comtypes.gen import Access
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QApplication, QMainWindow,  QMessageBox
```

Рисунок 13 – Используемые библиотеки

Пояснение для каждого модуля представлено в таблице 3.

Таблица 3 – Описание используемых библиотек

Модуль	Назначение
import os, shutil	Работа с файловой системой
from PyQt5 import QtCore, QtGui, QtWidgets from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox	Создание графического интерфейса пользователя
import sys	Позволяет работать с элементами среды выполнения Python, а именно, список аргументов командной строки
import cv2	Библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов
import numpy	Библиотека, добавляющая поддержку больших многомерных массивов и матриц

Продолжение таблицы 3.

import math	Предоставляет обширный функционал для работы с числами
from PIL import Image	Предоставляет поддержку при открытии, управлении и сохранении многих форматов изображений.
from scipy.spatial import distance	Модуль для подсчета расстояния между всеми точками в массиве
import matplotlib	Библиотека для визуализации двумерной графики
from comtypes.client import CreateObject from comtypes.gen import Access	Библиотека для создания и работы с БД Access

Таким образом, обозначены основные библиотеки для написания ПО. Библиотеки: os, shutil для работы с файловой системой; PyQt5 для создания графического интерфейса; cv2, numpy, Image, scipy и matplotlib для обработки изображения; comtypes для создания БД.

3.2. Реализация пользовательского интерфейса ПО

Создали формы для приложения в Qt Designer, далее конвертировали их в файлы с расширением *.py при помощи команды pyuic5 *.ui -o *.py.

Вид главного окна приложения представлено на рисунке 14.

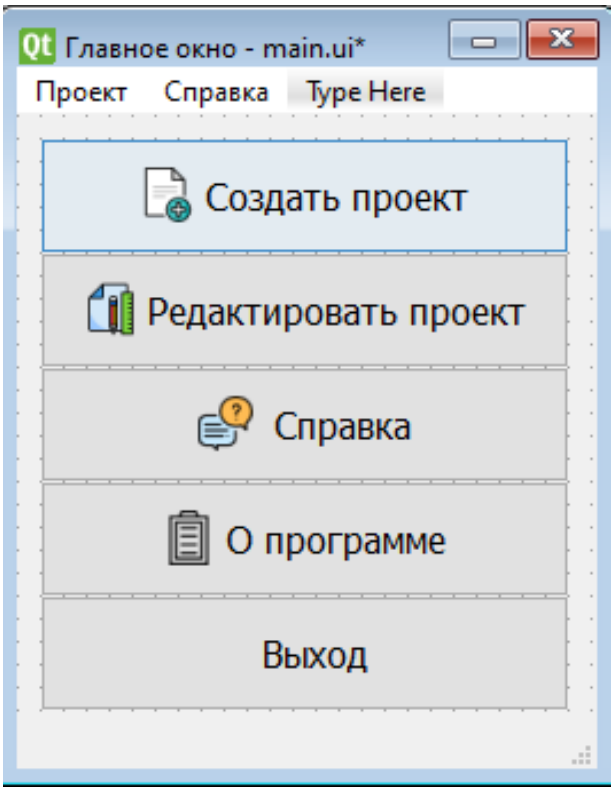


Рисунок 14 - Главное окно

Компоненты окна:

1) Кнопка создания проекта;
2) Кнопка редактирования проекта;
3) Кнопка «Справка» отображает руководство пользователя;
4) Кнопка о программе;
5) Навигационное меню, состоящее из 2 пунктов:

– «Проект» - отображает основные манипуляции с проектом. Содержит подпункт «Создать проект», «Редактировать проект» и «Выход»;

– «Помощь» - отображает информацию по работе с программой, необходимых пользователю. Содержит подпункты: «Справка» и «о программе.

Отрывок кода, содержащий функцию с наименованием объектов главного окна, представлен в листинге 1.

Листинг 1:

```
def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "Главное окно"))
    self.pushButton_2.setText(_translate("MainWindow", "Создать проект"))
    self.pushButton_3.setText(_translate("MainWindow", " Справка"))
    self.pushButton_4.setText(_translate("MainWindow", "О программе"))
    self.pushButton_5.setText(_translate("MainWindow", " Выход"))
    self.pushButton.setText(_translate("MainWindow", "Редактировать проект"))
    self.menu.setTitle(_translate("MainWindow", "Проект"))
    self.menu_2.setTitle(_translate("MainWindow", "Помощь"))
    self.action.setText(_translate("MainWindow", "Создать проект"))
    self.action_2.setText(_translate("MainWindow", "Выход"))
    self.action_3.setText(_translate("MainWindow", "Справка"))
    self.action_4.setText(_translate("MainWindow", "О программе"))
    self.action_5.setText(_translate("MainWindow", "Редактировать проект"))
```

Вид окна «Проект» представлен на рисунке 15.

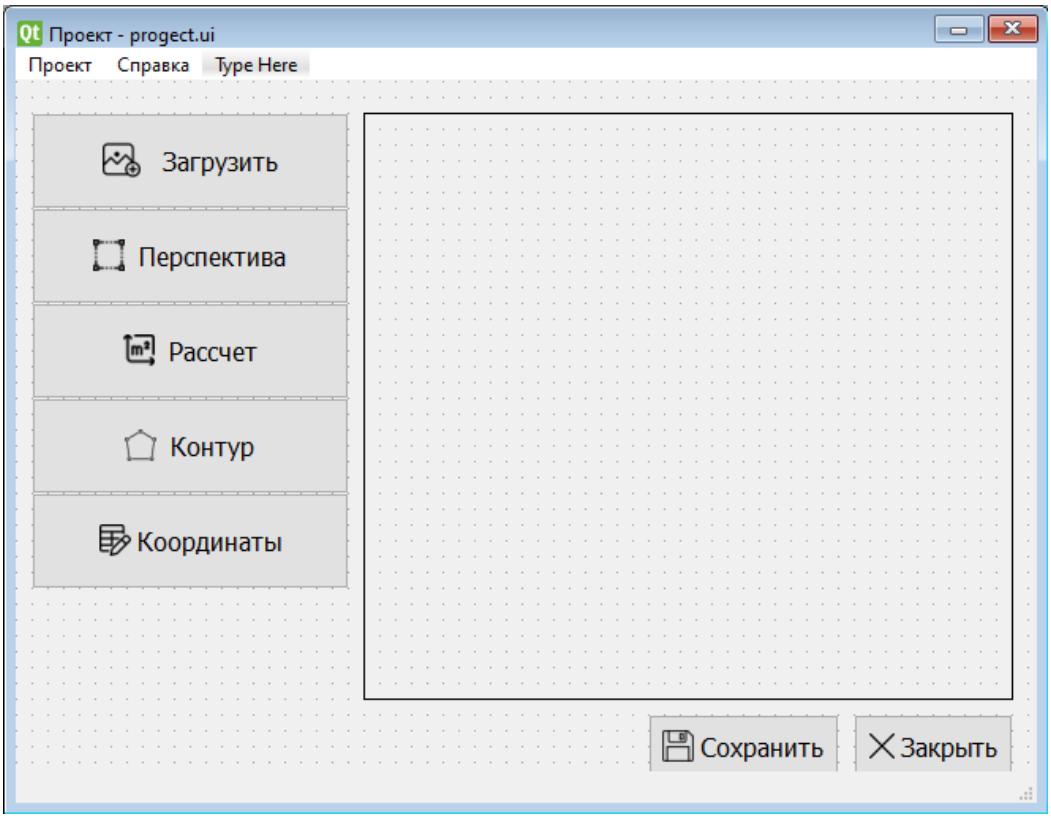


Рисунок 15 - Окно проект

Компоненты окна:

1) Кнопка загрузки фотографии;

2) Кнопка «Перспектива» для исправления перспективы;

3) Кнопка «Контур» для отображения и коррекции контура делового остатка;

4) Кнопка «Расчет» для вывода площади делового остатка;

5) Кнопка «Координаты» для отображения базы данных с координатами делового остатка;

6) Кнопка «Сохранить» для сохранения проекта;

7) Кнопка «Закрыть» для закрытия окна;

8) Навигационное меню, состоящее из 2 пунктов:

– «Проект» - содержит подпункт «Сохранить» и «Выход»;

– «Помощь» - отображает информацию по работе с программой, необходимых пользователю. Содержит подпункты: «Справка» и «о программе.

Отрывок кода, содержащий функцию с наименованием объектов окна проект, представлен в листинге 2.

Листинг 2:

```
def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "Проект"))
    self.perspektiva.setToolTip(
      _translate("MainWindow", "<html><head/><body><p>Исправить изображение</p></body></html>"))
    self.perspektiva.setText(_translate("MainWindow", " Перспектива"))
    self.raschet.setToolTip(_translate("MainWindow", "<html><head/><body><p>Площадь детали</p></body></html>"))
    self.raschet.setWhatsThis(
      _translate("MainWindow", "<html><head/><body><p>Рассчитать площадь детали</p></body></html>"))
    self.raschet.setText(_translate("MainWindow", " Расчет"))
    self.contur.setToolTip(
      _translate("MainWindow", "<html><head/><body><p>Вывести контур детали</p></body></html>"))
    self.contur.setText(_translate("MainWindow", " Контур"))
    self.conttable.setToolTip(
      _translate("MainWindow", "<html><head/><body><p>Открыть базу данных с координатами</p></body></html>"))
```

```
self.conttable.setText(_translate("MainWindow", "Координаты"))
self.zakryt.setText(_translate("MainWindow", "Закрыть"))
self.saveprogekt.setText(_translate("MainWindow", "Сохранить"))
self.pushButton.setText(_translate("MainWindow", "  Загрузить"))
self.menu.setTitle(_translate("MainWindow", "Проект"))
self.menu_2.setTitle(_translate("MainWindow", "Помощь"))
self.action.setText(_translate("MainWindow", "Сохранить"))
self.action_2.setText(_translate("MainWindow", "Выход"))
self.action_3.setText(_translate("MainWindow", "Справка"))
self.action_4.setText(_translate("MainWindow", "О программе"))
```

Окно «Редактировать проект» содержит компоненты окна проект. Вид окна представлен на рисунке 16.

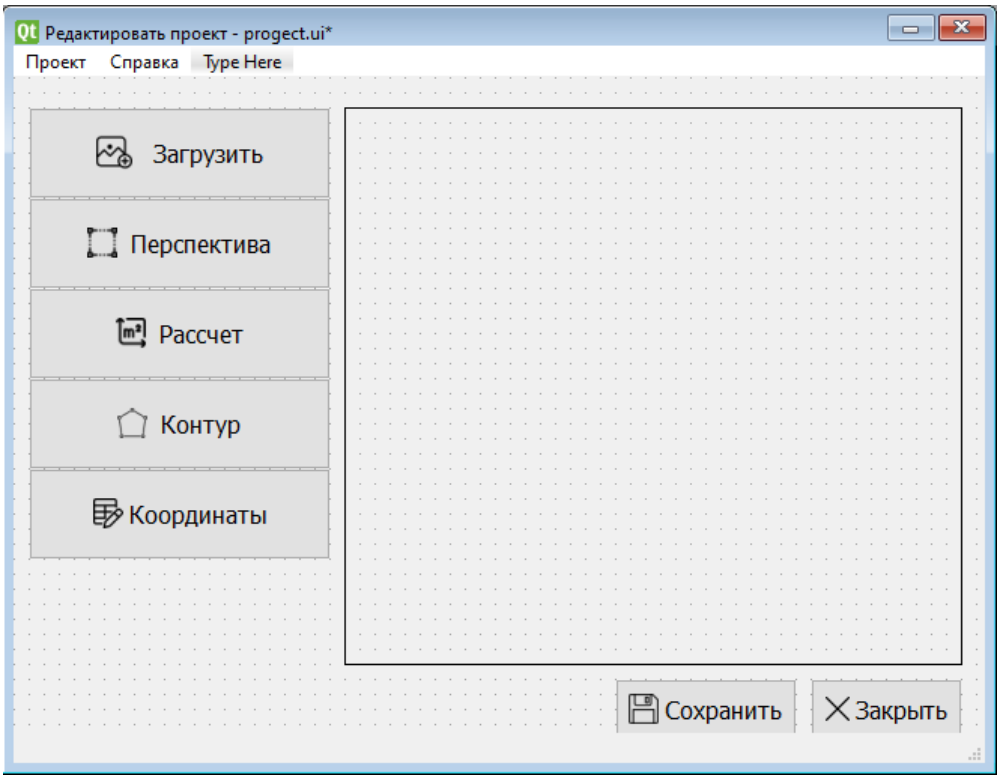


Рисунок 16 - Редактировать проект

Вид окна «О программе» представлен на рисунке 17. Содержит основную информацию о программе.

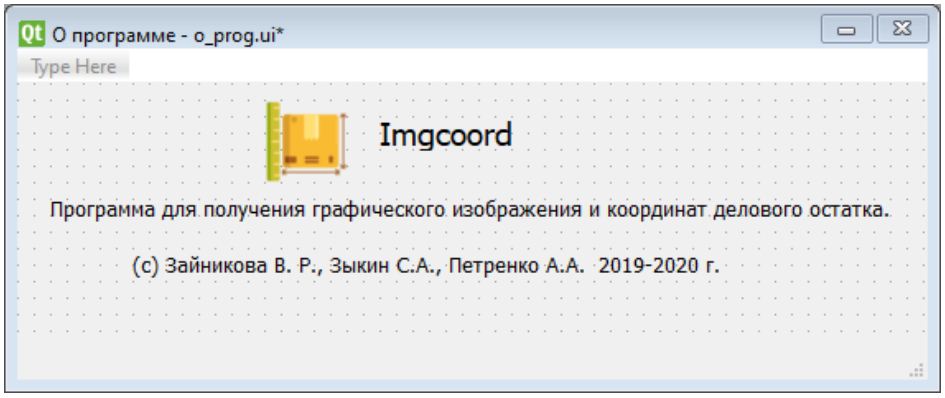


Рисунок 17 - О программе

Отрывок кода, содержащий функцию с наименованием объектов окна, представлен в листинге 3.

Листинг 3:

```
def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "О программе"))
    self.label.setText(_translate("MainWindow", "(c) Зайникова В. Р., Зыкин С. А., Петренко А. А.  2019-2020 г. "))
    self.label_2.setText(
      _translate("MainWindow", "Программное обеспечение получения размеров делового остатка нестандартной формы"))
    self.label_3.setText(_translate("MainWindow", "Imgcoord"))
    self.label_5.setText(_translate("MainWindow", "по цифровой фотографии."))
```

Таким образом, реализован пользовательский интерфейс приложения, который состоит из главного окна, окон для создания и редактирования проекта, а так же формой содержащей основную информацию о программе

3.3. Сведения об ограничениях программы

Программа «Imgcoord» имеет следующие ограничения:

1) Единицы измерения – миллиметры;
2) Путь к папке проекта не должен содержать русский алфавит;
3) Путь к изображению ДО не должен содержать русский алфавит;
4) Программа находит только внешний контур ДО, отверстия не учитываются;
5) Лист формата А4 должен располагаться, на фотографии, вертикально;
6) Фон фотографии должен быть однотонного цвета, если цвет делового остатка светлый, то фон темный, если цвет остатка темный, то фон светлый.

Фотография, загружаемая пользователем, должна соответствовать ранее обозначенным ограничениям.

При нахождении контура ДО, он находится с небольшой погрешностью, обусловленной округлением данных.

Таким образом, обозначены основные ограничения и условия работы системы.

3.4. Описание основных функций системы

При реализации системы создали глобальные переменные, необходимые для работоспособности программы:

Переменные F и F1, которые соответственно содержат путь для создания проекта и путь для редактирования проекта.

n_points=[] – массив для хранения координат контура.

Для корректной работы программы, введены условия, при которых для редакции проекта выбирается только та папка, в которой содержится проект, иначе выходит сообщение об ошибке. Так же отрывается окно ошибки, если путь к папке или к изображению не указан.

Вид окна ошибки, при некорректном выборе папки содержащей проект, представлен на рисунке 18.

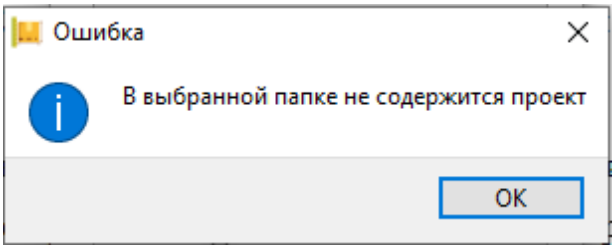


Рисунок 18 – Ошибка при неправильно выбранной папке

Вид окна ошибки, если путь к проекту не указан, представлен на рисунке 19.

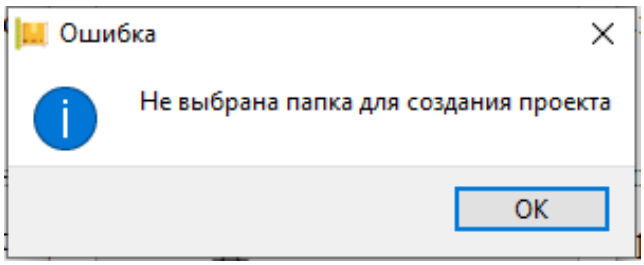


Рисунок 19 – Окно ошибки при неукозании пути к проекту

Вид окна ошибки, если путь к изображению не указан, представлен на рисунке 20.

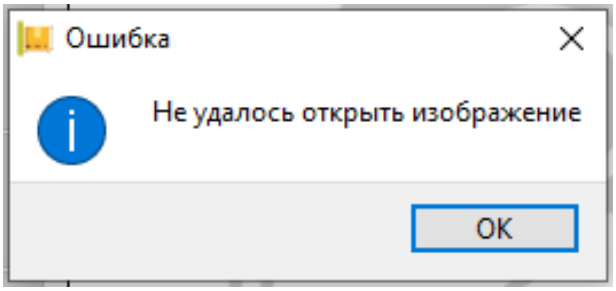


Рисунок 20 – Окно ошибки при неукозании пути к изображению

Таким образом, описали глобальные переменные приложения. Реализовали корректное открытие папки содержащей проект, а так же описали события, при отсутствии выбора пути к папке и изображеню.

3.2.1. Создание проекта

При запуске приложения открывается главное окно представленое на рисунке 21.

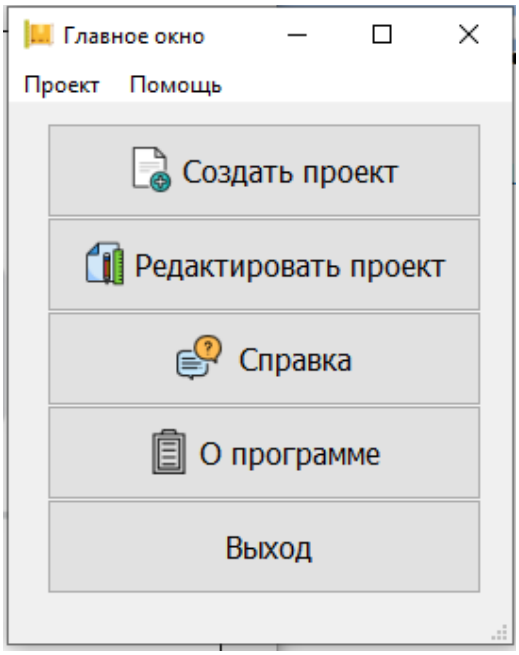


Рисунок 21 – Главное окно программы

При нажатии на кнопку «Создать проект» открывается диалоговое окно, в котором необходимо выбрать папку хранения для проекта. Фрагмент кода, отвечающий за открытие диалогового окна, представлен в листинге 4.

Листинг 4:

```
def show_window_2(self):
        global F
        F = QtWidgets.QFileDialog.getExistingDirectory(self, "Создать проект в папке", ".")
        if F == "":
            QtWidgets.QMessageBox.information(self, "Ошибка",
                                              "Не выбрана папка для создания проекта")
        else:
            folder = F
            for the_file in os.listdir(folder):
                if the_file == 'open_img.jpg':
                    os.remove(F + '/open_img.jpg')
                if the_file == 'size_img.jpg':
                    os.remove(F + '/size_img.jpg')
                if the_file == 'transform_img.jpg':
                    os.remove(F + '/transform_img.jpg')
            self.w2 = Window2()
            self.w2.show()

        folder = 'temp/'
        for the_file in os.listdir(folder):
            file_path = os.path.join(folder, the_file)
            if os.path.isfile(file_path):
                os.unlink(file_path)
```

Диалоговое окно представлено на рисунке 22.

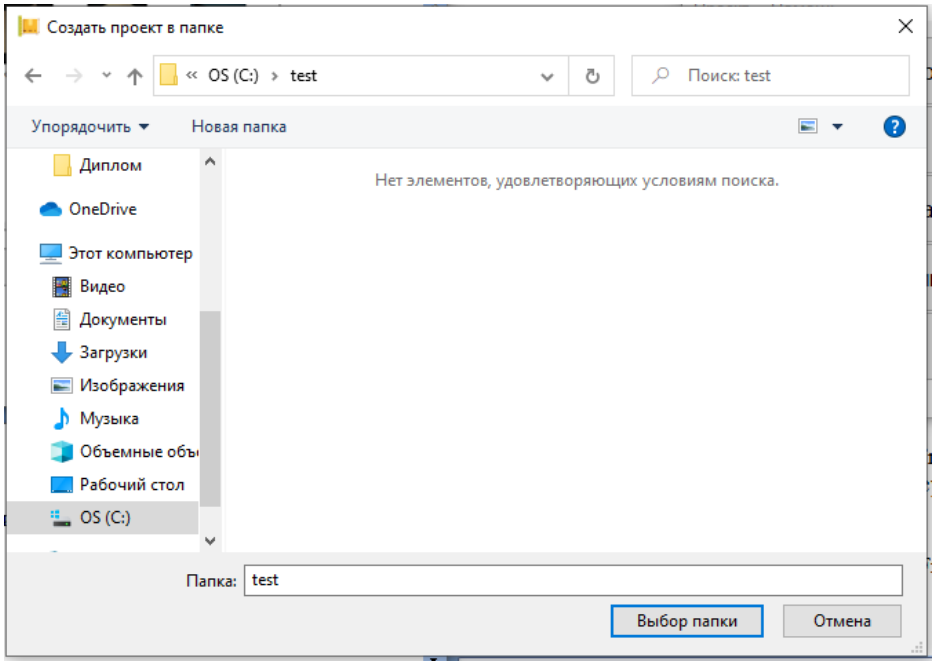


Рисунок 22 – Диалоговое окно

Таким образом, реализовали функцию создания проекта при нажатии на кнопку «Создать проект».

3.2.2. Загрузка изображения

При открытии окна «Проект», доступна только кнопка загрузки изображения, пока фотография не выбрана, остальные кнопки для взаимодействия с изображением, неактивны.

Вид окна «Проект» при запуске представлен на рисунке 23.

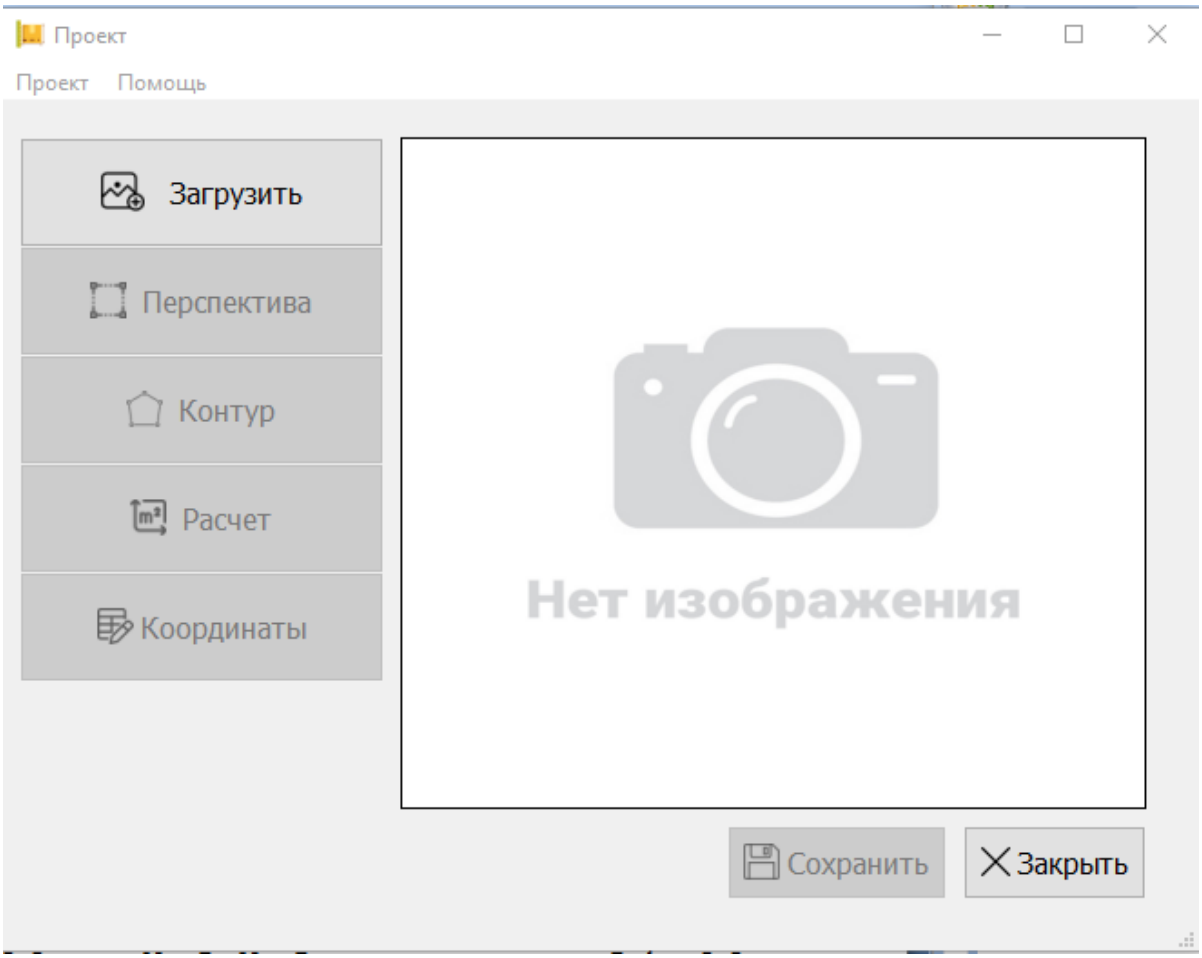


Рисунок 23 – Запуск окна «Проект»

При нажатии на кнопку «Загрузить» открывается диалоговое окно, в котором необходимо выбрать фото ДО. В случае если фото загружено, изображение копируется в папку temp под именем open_img. Фрагмент кода, отвечающий за открытие диалогового окна, представлен в листинге 5.

Листинг 5:

```
def view_img(self):
    fileName = self.loag_img()
    if fileName == "":
        QtWidgets.QMessageBox.information(self, "Ошибка",
                                          "Не удалось открыть изображение")
    else:
        img = cv2.imread(fileName)
        cv2.imwrite('temp/open_img.jpg', img)
        self.image.setPixmap(QtGui.QPixmap("temp/open_img.jpg"))
        self.perspektiva.setEnabled(True)
        self.contur.setEnabled(False)
        self.raschet.setEnabled(False)
        self.conttable.setEnabled(False)
        self.saveprogekt.setEnabled(False)
        self.action.setEnabled(False)
        pass
```

Диалоговое окно представлено на рисунке 24.

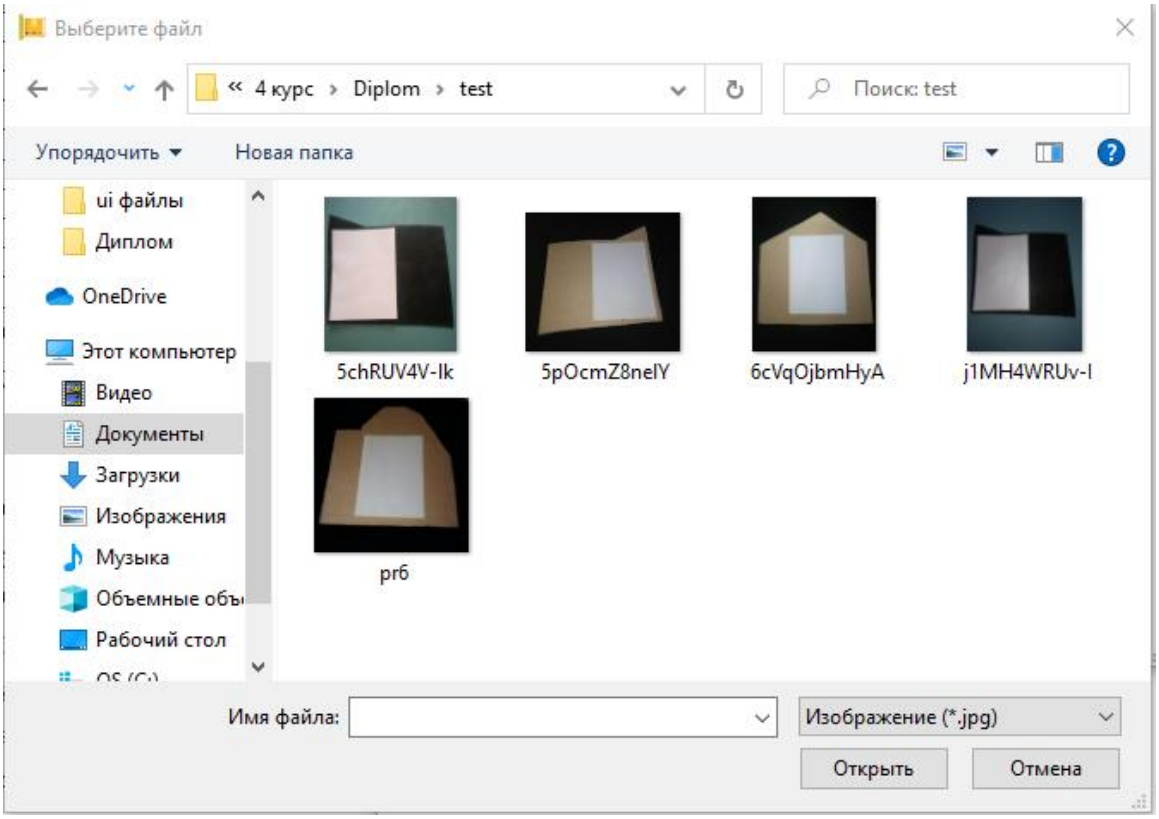


Рисунок 24 – Диалоговое окно

Таким образом, при нажатии на кнопу «Загрузить» откроется диалговое окно, в котором необходимо указать путь к фотографии с изображением ДО.

3.2.3. Исправление перспективы изображения

После загрузки изображения активируется кнопка «Перспектива». При нажатии на ее открывается окно, в котором необходимо область, для исправления перспективы.

Для этого ДО необходимо вписать в прямоугольник относительно листа А4.

Фрагмент кода, отвечающий за выделение области и открытия окна для исправления перспективы, представлен в листинге 6.

Листинг 6:

```
def get_contour(self, rescaled_image):
    IM_HEIGHT, IM_WIDTH, _ = rescaled_image.shape
    TOP_RIGHT = (IM_WIDTH - 50, 50)
    BOTTOM_RIGHT = (IM_WIDTH - 50, IM_HEIGHT - 50)
    BOTTOM_LEFT = (50, IM_HEIGHT - 50)
    TOP_LEFT = (50, 50)
    screenCnt = np.array([[TOP_RIGHT], [BOTTOM_RIGHT], [BOTTOM_LEFT], [TOP_LEFT]])
    return screenCnt.reshape(4, 2)

def interactive_get_contour(self, screenCnt, open_image):
    poly = Polygon(screenCnt, animated=True,  fill=False, color="blue")
    fig, ax = plt.subplots()
    ax.add_patch(poly)
    ax.set_title(('Выделите область для коррекции перспективы делового остатка. \n'
                  'Закройте окно, когда закончите.'))
    p = PolygonInteractor(ax, poly)
    plt.imshow(open_image)
    plt.show()
    new_points = p.get_poly_points()[:4]
    new_points = np.array([[p] for p in new_points], dtype="int32")
    return new_points.reshape(4, 2)
```

Окно для выделения ДО предоставлено на рисунке 25.

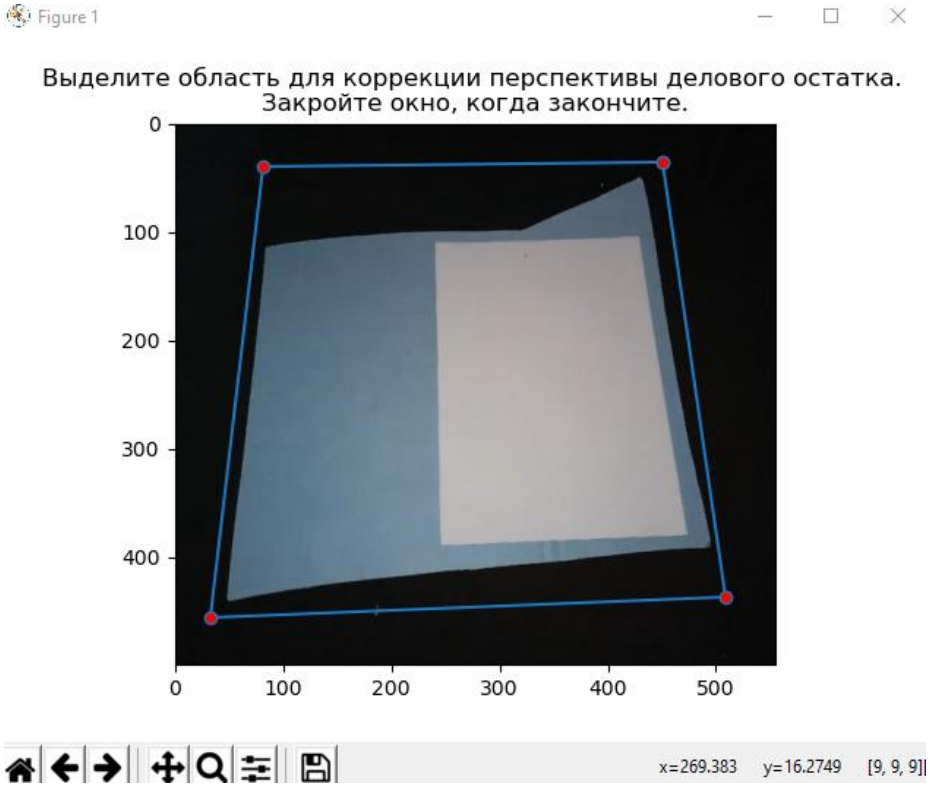


Рисунок 25 – Выделение области для исправления перспективы

Окно с результатом исправления перспективы предоставлено на рисунке 26.

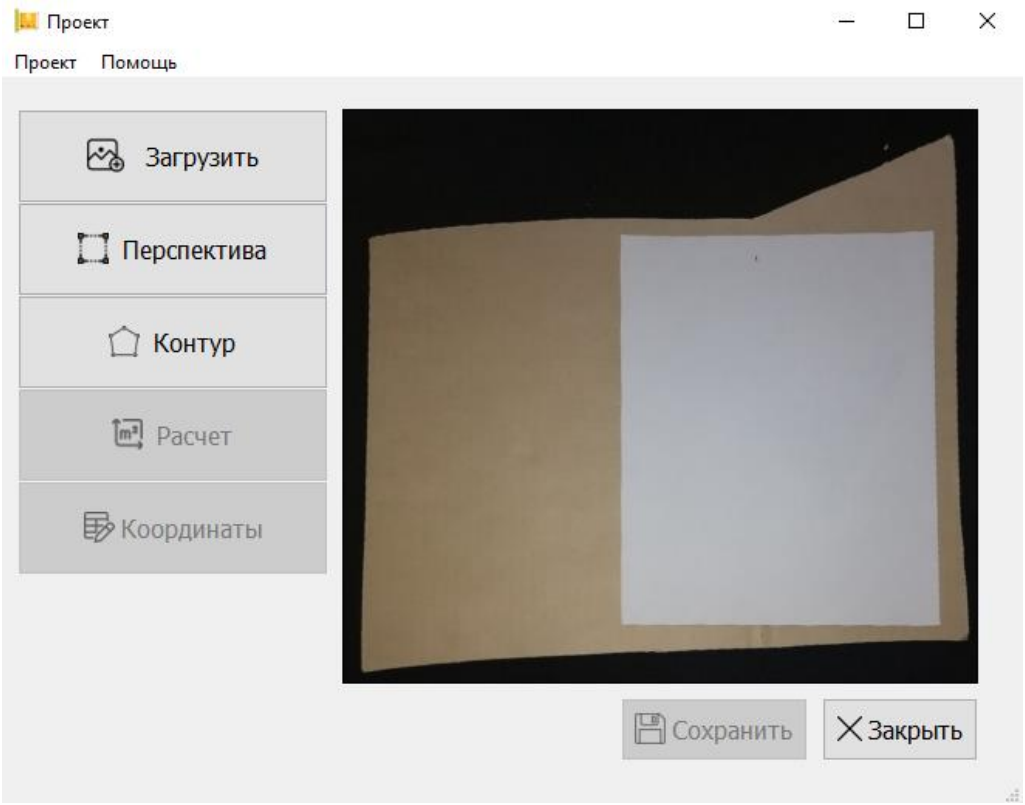


Рисунок 26 – Выделение области для исправления перспективы

Часть кода, отвечающая за исправление перспективы, представлена в листинге 7.

Листинг 7:

```
    def four_point_transform(self, image, pts):
        rect = self.order_points(pts)
        (tl, tr, br, bl) = rect
        widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
        widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
        maxWidth = max(int(widthA), int(widthB))
        heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
        heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
        maxHeight = max(int(heightA), int(heightB))
        dst = np.array([
            [0, 0],
            [maxWidth - 1, 0],
            [maxWidth - 1, maxHeight - 1],
            [0, maxHeight - 1]], dtype="float32")
        M = cv2.getPerspectiveTransform(rect, dst)
        warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))
        return warped

    def obrabotka(self, image_open):

        RESCALED_HEIGHT = 500.0
        # загрузить изображение и вычислить отношение старой высоты к новой высоте, клонировать и изменить его размер
        image = cv2.imread(image_open)
        assert (image is not None)
        ratio = image.shape[0] / RESCALED_HEIGHT
        orig = image.copy()
        rescaled_image = self.resize(image, height=int(RESCALED_HEIGHT))
        # получить контур
        screenCnt = self.get_contour(rescaled_image)
        screenCnt = self.interactive_get_contour(screenCnt, rescaled_image)
        # применить коррекцию перспективы
        warped = self.four_point_transform(orig, screenCnt * ratio)
        # сохранить преобразованное изображение
        cv2.imwrite('temp/transform_img.jpg', warped)
```

Но если перспектива исправлена некорректно отроется окно ошибки, представленное на рисунке 27.

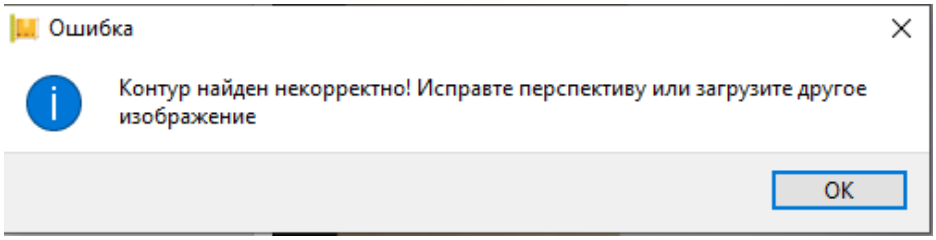


Рисунок 27 – Окно ошибки

Таким образом, после преобразования перспективы, исправленное изображение под именем transform_img копируется в папку temp.

3.2.4. Нахождение контура делового остатка

После исправления перспективы активируется кнопка «Контур». При нажатии на ее открывается окно, в котором при необходимости можно откорректировать найденный контур.

Окно с найденным контуром предоставлено на рисунке 28.

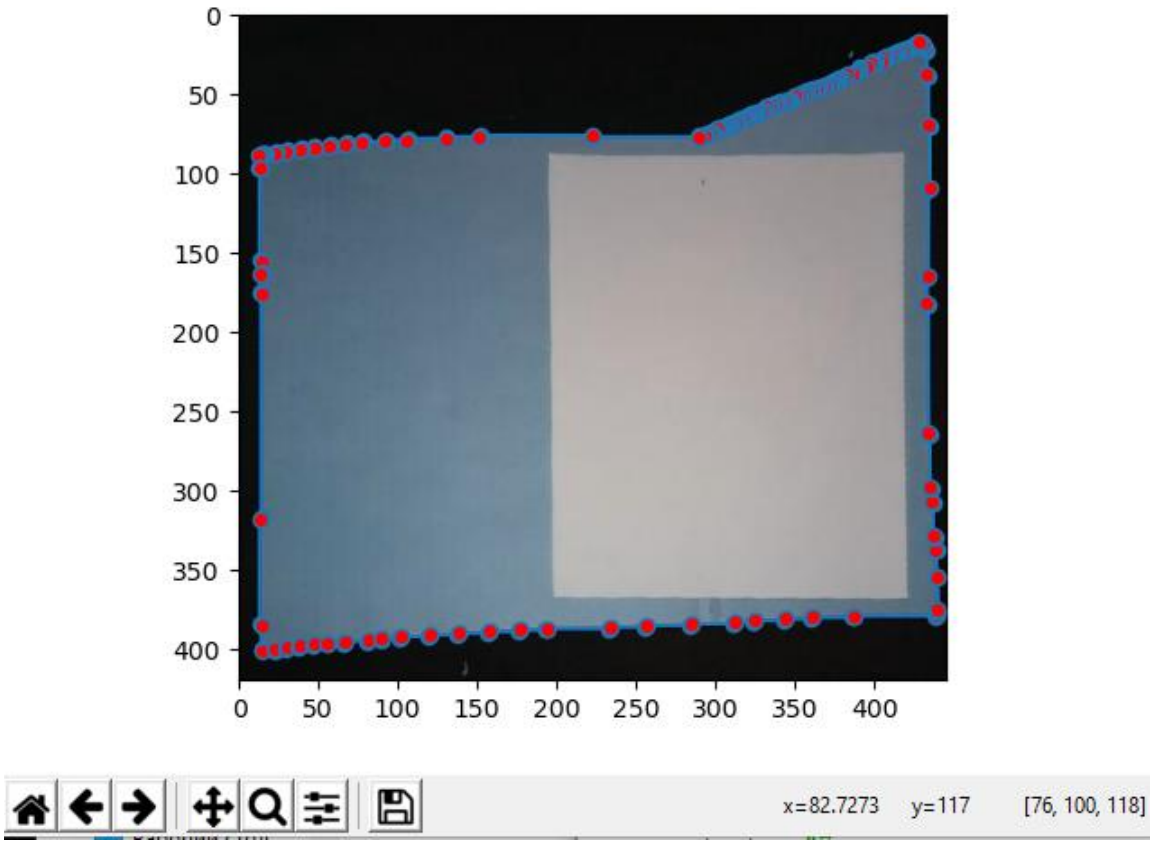


Рисунок 28 – Контур ДО

Код нахождения контура представлен в листинге 8:

Листинг 8:

```
image = cv2.imread("temp/size_img.jpg")
edged = cv2.Canny(image, 10, 250)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7))
closed = cv2.morphologyEx(edged, cv2.MORPH_CLOSE, kernel)
sharpen = cv2.GaussianBlur(closed, (0, 0), 3)
sharpen = cv2.addWeighted(closed, 1.5, sharpen, -0.5, 0)
cnts, _ = cv2.findContours(sharpen, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
# найдем самый большой контур
c = sorted(cnts, key=cv2.contourArea, reverse=True)[0]
array_x = []
array_y = []
        for i in range(len(c) - 1):
            xy = c[i][0]
            array_x.append(xy[0])
            array_y.append(xy[1])

        poly = Polygon(list(zip(array_x, array_y)), animated=True, fill=False, color="blue")
        fig, ax = plt.subplots()
        ax.add_patch(poly)
        p = PolygonInteractor(ax, poly)
        plt.imshow(image)
        plt.show()
        new_points = p.get_poly_points()
        new_points = np.array([[p] for p in new_points])
        global n_points
        n_points=new_points
```

Таким образом, после нахождения контура, исправленное изображение под именем size_img копируется в папку temp.

3.2.5. Нахождение площади и внесение координат в базу данных

После нахождения контура активируются кнопки «Расчет» и «Координаты».

Реализована возможность находжения площади ДО. При нажатии на кнопку «Расчет» отрывается диалоговое окно содержащее сообщение о площади ДО.

Окно, содержащее сообщение о площади предоставлено на рисунке 29.

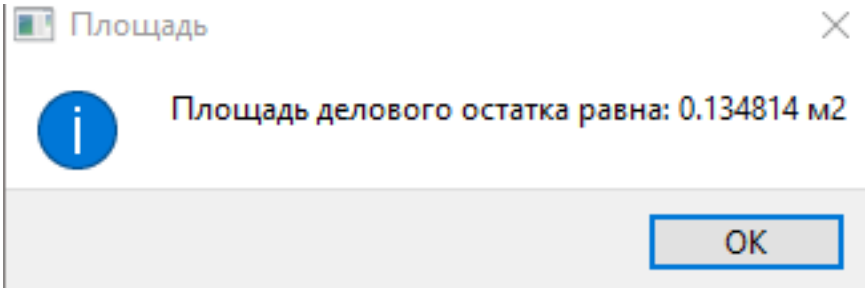


Рисунок 29 – Площадь ДО

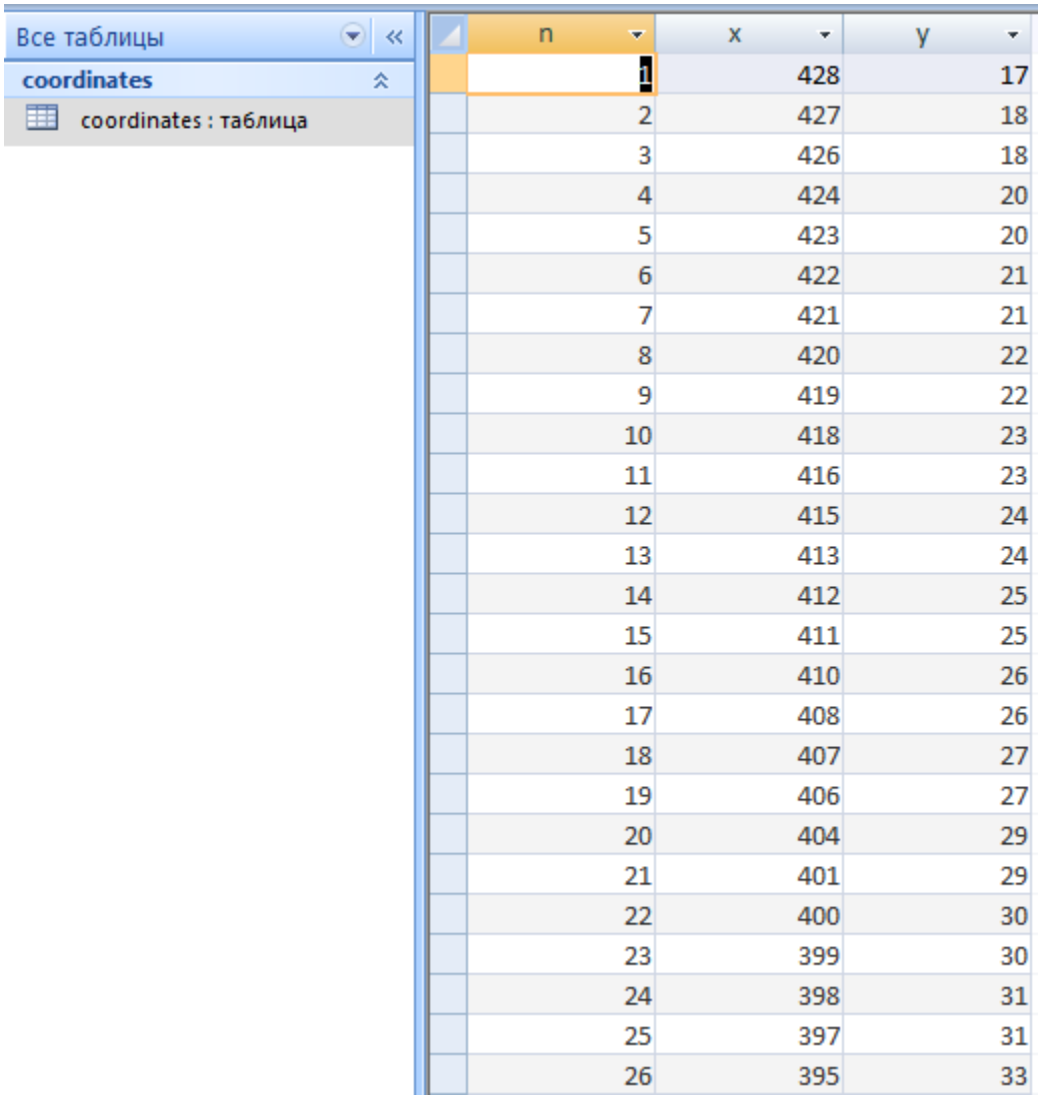
Код открытия диалогового окна представлен в листинге 9.

Листинг 9:

```
def find_cntr(self):
        image = cv2.imread("temp/size_img.jpg")
        edged = cv2.Canny(image, 10, 250)
        kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7))
        closed = cv2.morphologyEx(edged, cv2.MORPH_CLOSE, kernel)
        sharpen = cv2.GaussianBlur(closed, (0, 0), 3)
        sharpen = cv2.addWeighted(closed, 1.5, sharpen, -0.5, 0)
        cnts, _ = cv2.findContours(sharpen, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        # найдем самый большой контур
        c = sorted(cnts, key=cv2.contourArea, reverse=True)[0]
        sq = cv2.contourArea(c)
        s = sq / 1000000
        text = "Площадь делового остатка равна: "
        text = (text) + str(s) + " м2"
        msgBox = QMessageBox()
        msgBox.setIcon(QMessageBox.Information)
        msgBox.setText(text)
        msgBox.setWindowTitle("Площадь")
        returnValue = msgBox.exec()
```

Так же реализована возможность внести координаты ДО в БД Access. При нажатии на кнопку «Координаты» в папку, куда сохранен проект,

автоматически создается БД, которая содержит все координаты ДО (см. рисунок. 30).



n	x	y
1	428	17
2	427	18
3	426	18
4	424	20
5	423	20
6	422	21
7	421	21
8	420	22
9	419	22
10	418	23
11	416	23
12	415	24
13	413	24
14	412	25
15	411	25
16	410	26
17	408	26
18	407	27
19	406	27
20	404	29
21	401	29
22	400	30
23	399	30
24	398	31
25	397	31
26	395	33

Рисунок 30 – Координаты ДО

Код внесения координат БД представлен в листинге 10.

Листинг 10:

```
# найдем самый большой контур
 c = sorted(cnts, key=cv2.contourArea, reverse=True)[0]
array_x = []
array_y = []
for i in range(len(c) - 1):
xy = c[i][0]
array_x.append(xy[0])
array_y.append(xy[1])

def create_database(self):
```

```
            global F
            global n_points
            access = CreateObject('Access.Application')
            DBEngine = access.DBEngine
            db = DBEngine.CreateDatabase((F+'/coordinates.mdb'),
Access.DB_LANG_GENERAL)

            db.BeginTrans()

            db.Execute("CREATE TABLE coordinates (n Integer, x Integer, y
Integer)")

            for i in range(len(n_points)):
                xy = n_points[i][0]
                x = math.ceil(xy[0])
                y = math.ceil(xy[1])
                n = i + 1
                db.Execute("INSERT INTO coordinates VALUES (" + str(n) + "," +
str(x) + "," + str(y) + ")")
            db.CommitTrans()
            db.Close()
```

Так же после внесения координат в базу данных, активируются кнопка «Сохранить». После нажатия на нее, отрывается диалоговое окно с сообщением о сохранении проекта представленное на рисунке 31.

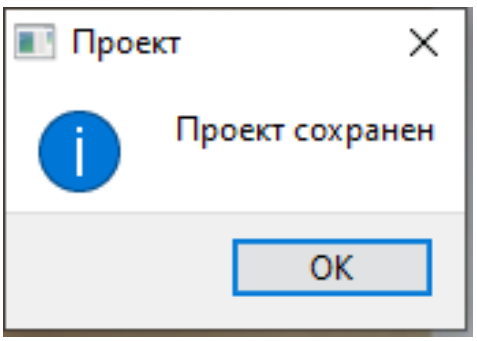


Рисунок 31 – Координаты ДО

Таким образом, реализованы функции нахождения площади ДО и внесения координат в БД. Так же реализована функция сохранения проекта.

3.2.6. Редактирование проекта

При нажатии на кнопку «Редактировать проект» открывается диалоговое окно, в котором необходимо выбрать папку в которой хранится проект. Фрагмент кода, отвечающий за открытие диалогового окна, представлен в листинге 11.

Листинг 11:

```
def show_window_3(self):
    global F1
    image1 = "no"
    image2 = "no"
    image3 = "no"
    F1 = QtWidgets.QFileDialog.getExistingDirectory(self, "Открыть проект", ".")
    if F1 == "":
        QtWidgets.QMessageBox.information(self, "Ошибка",
                                          "Не выбрана папка для редактирования проекта")
    else:
        files = os.listdir(F1)
        for the_file in files:
            if the_file == 'open_img.jpg':
                image1 = "yes"
            if the_file == 'size_img.jpg':
                image2 = "yes"
            if the_file == 'transform_img.jpg':
                image3 = "yes"
        if (image1 == 'yes' and image2 == 'yes' and image3 == 'yes'):
            self.w3 = Window3()
            self.w3.show()
        else:
            QtWidgets.QMessageBox.information(self, "Ошибка",
                                              "В выбранной папке не содержится проект")
    folder = 'temp/'
    for the_file in os.listdir(folder):
        file_path = os.path.join(folder, the_file)
        if os.path.isfile(file_path):
            os.unlink(file_path)
```

Диалоговое окно представлено на рисунке 32.

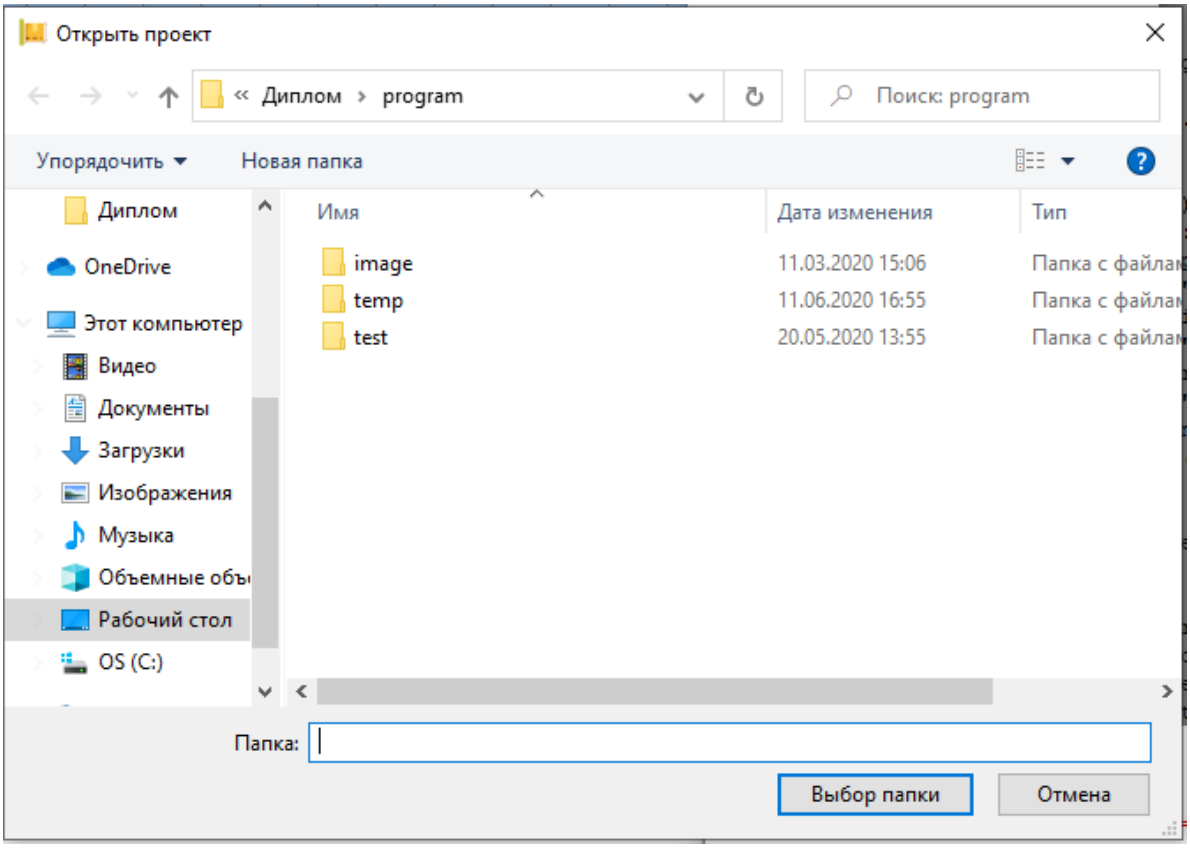


Рисунок 32 – Диалоговое окно

Таким образом, реализовали функцию открытия папки для редактирования проекта при нажатии на кнопку «Редактировать проект».

Вывод:

В результате реализованы основные функции для работы приложения с использованием библиотек. Разработан пользовательский интерфейс. Созданы формы, выполняющие задачи и функции, определенные из анализа предметной области.

4. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

Эффективность любого специализированного программного обеспечения определяется его назначением, результатами использования по этому назначению, а также затратами на его создание и последующую эксплуатацию.

Целью экономической части является расчёт эффективности от внедрения автоматизированной системы учета техники в сервисном центре.

Основной задачей данного раздела является определение целесообразности разработки и величины экономического эффекта от внедрения программного продукта.

Внедрение системы учета техники в сервисном центре позволит получить следующие эффекты:

1. Повышение качества и скорости обслуживания клиентов при приеме и выдаче неисправного оборудования.
2. Рост производительности сотрудников при приеме и выдаче.
3. Повышение прибыли предприятия за счет сокращения труда выполняемого человеком.

Для определения целесообразности разработки и величины экономического эффекта от внедрения программного продукта необходимо провести расчёт технико-экономических показателей.

4.1. Расчет трудоемкости разработки

Расчет трудоемкости разработки ПО – стоимость потраченного времени разработчиком на разработку ПО. Она определяется формулой:

$$C = З * Т, \quad (8)$$

где З – среднедневная стоимость работ, денежные единицы, руб.;

Т – общая трудоемкость проекта, дни.

Стоимость работы одного часа составляет 150 руб.

З = 150 * 8 = 1200 рублей в день.

Для разработки ПО потребовалось 30 рабочих дня.

С = 30 * 1200 = 36000 рублей.

4.2. Определение плановой себестоимости проведения работ

В себестоимость проведения работ включается стоимость оборудования и монтажа.

Ц = СО + М (9)

Для установки и работы системы не требуется покупка нового оборудования. Поэтому, стоимость проведения работ равна нулю.

Ц = 0 рублей.

4.3. Экономический эффект

Экономический эффект от использования программного продукта выражается в экономии затрат времени на обработку данных.

Рассчитав затраты на трудоемкость и себестоимость получим полную цену за работающую систему:

Ц полн = Ц + С = 36000 + 0 = 36000 рублей.

Для расчета экономии возьмем стоимость работы одного часа сотрудника – 90 руб., среднее количество рабочих дней в месяце – 22 день.

Рассчитаем месячные расходы на оплату труда:

Змес = 90 * 8 * 22 = 15840 руб.

Годовые расходы на оплату труда:

Згод = 15840 * 12 = 190080 руб.

Без использования автоматизированной системы учета техники, время создания сохранной расписки и выдача оборудования одному клиенту составляет 7 минут. С использованием автоматизированной системы время составляет одну минуту. В течении дня сервисный центр обслуживает около 25 клиентов.

Найдем экономию времени на выполнение задач сотрудника.

Рассчитаем месячные расходы на оплату труда без использования АС:

Змес = 90 * 7/60 * 25*30 = 7875 руб.

Рассчитаем месячные расходы на оплату труда при использовании АС:

Змес = 90 * 1/60 * 25*30 = 1125 руб.

Итого ежемесячная экономия составит:

Эмес =7875-1125=6750 руб.

Ежегодная экономия составит:

Эгод =6750*12=81000 руб.

При стоимости системы 36000 руб. её установка окупится на шестом месяце работы, а ежемесячная экономия составит 6750 руб.

Вывод

Рассчитана полная цена за работающую систему, которая равна $Ц_{полн}$ = 36000 рублей. Её установка окупится на шестом месяце работы, а ежемесячная экономия составит 6750 руб.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы была проанализирована предметная область. Обозначены действующее лицо и функционал для моделирования и реализации ПО, получения контура с реальными размерами и площадью делового остатка

Для проектирования и реализации ПО получения контура с реальными размерами и площадью делового остатка, создана модель AS-IS, методом SADT, для реализации программы.

Разработана иаематическая модель нахождения контура ДО нестандартной формы по цифровой фотографии с помощь. языка программирования Python, с использованием библиотеки компьютерного зрения OpenCV.

Разработан пользовательский интерфейс и описаны основные функции, использованные при программировании приложения.

Рассчитано экономическое обоснование. В ходе, которого были вычислены трудоемкость разработки ПО, себестоимость проведения работ и экономический эффект.

Для нахождения площади и контура с реальными размерами делового остатка, была разработана программа «Imgcoord».

Поставленные цели и задачи были полностью выполнены.

Перспективой для дальнейших исследований является доработка программного обеспечения. Реализовать возможность выбирать шаблон для приведения изображения к реальным размерам, используя не только лист А4. Эта функция позволит работать с ДО любых размеров.

Список использованных источников

1. Нанавова Т.А. Алгоритм извлечения текста из видео с использованием библиотеки компьютерного зрения OPENCV // Ростовский научный журнал. – 2016. – № 7. – С. 21–40.
2. Михалин Д.А., Белов Ю.С. Применение библиотеки OPENCV для задачи распознавания лиц // Актуальные проблемы авиации и космонавтики. – 2016. – № 12. – С. 558–559.
3. Назарова Т.Ю., Лавров Д.Н. Компьютерное моделирование идентификации личности по радужной оболочке глаза на основе OPENCV // Математические структуры и моделирование. – 2014. – № 1(29). – С. 43–64.
4. Тихонова Т.С., Белов Ю.С. Основные подходы к отслеживанию и распознаванию лица // Электронный журнал: наука, техника и образование. – 2016. – № 2(6). – С. 111–115.
5. R. Lienhart, A. Kuranov, V. Pisarevsky. Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. MRL Technical Report, 2002
6. Л.Н. Чабан Методы и алгоритмы распознавания образов в автоматизированном дешифрировании данных дистанционного зондирования: учебное пособие. – М.: МИИГАиК, 2016, – 94 с.
7. Bradsky G., Kaehler A. Learning OpenCV — O'Reilly, 2008. — С. 571
8. Нгуен К.М., Колючкин В.Я. Алгоритмы контурной сегментации и распознавания образов объектов систем технического зрения // наука и образование: научное издание МГТУ им. Н.Э. Баумана. – 2013. – № 4. – С. 187–200.
9. J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. CoRR, abs/1506.02640, 2015.
10.] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CoRR, abs/1311.2524, 2013.
11. R. B. Girshick. Fast R-CNN. CoRR, abs/1504.08083, 2015

12.] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. CoRR, abs/1506.01497, 2015.

13. Kainz, O ; Horecny, MW ; Jakab, F ; Cymbalak, D. Estimating the Object Size from Static 2D Image CoRR, 2015, pp. 15-17.

14. Яне Б. Цифровая обработка изображений : пер. с англ. М.: Техносфера, 2007. 584 с

15. Chang PL, Teng WG. 2007. Exploiting the self-organizing map for medical image segmentation in Computer-Based Medical Systems, Twentieth IEEE International Symposium on, pp. 281288.

16. Zhou ZJ,Wu H. 2010. Digital Image Processing: Oart 1. Ventus Publishing ApS

17. Canny J. E. A computational approach to edge detection // IEEE Trans Pattern Analysis and Machine Intelligence. 1986. № 8. P. 679 – 698.

18. Цветков А. А., Шорох Д. К., Зубарева М. Г., Юрсков С. В., Шуклин А. В., Хамуш А. Л., Ануфриев И. Б. Алгоритмы распознавания объектов // Технические науки: проблемы и перспективы. – 2016. –С. 20–28.

19. Шитова О.В. Пухляк А.Н. Дроб Е.М. Анализ методов сегментации текстурных областей изображений в системах обработки изображений // Научные ведомости белгородского государственного университета. серия: экономика. информатика. – 2014. – № 8(179). – С. 182–188.

20. Родзин С.И., Родзина О.Н., Эль-Хатиб С.А. Гибридный муравьиный алгоритм сегментации медицинских изображений // Вестник Чувашского университета. – 2017. – № 3. – С. 262–272.

21. В.С. Комков, М.В. Дюдин Метод сегментации растровых изображений на основе выбора оптимальных направлений обработки пикселей изображения // Российская наука и образование сегодня: проблемы и перспективы. – 2015. – № 2(5). – С. 151–154.

22. Шапиро Л., Стокман Дж. Компьютерное зрение. М. : Бином, 2006. 752 с.

23. Гонзалес Р., Вудс Р. Цифровая обработка изображений. – М.: Техносфера, 2006. – 1072с

24. Rosenfeld A., Kak A.C. Digital Picture Processing. New York : Academic press, 1976.

25. Стругайло В.В. Обзор методов фильтрации и сегментации цифровых изображений // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2012. № 5.

26. ГОСТ 166-89. Межгосударственный стандарт. Штангенциркули. Технические условия // Доступ из справ.-правовой системы КонсультантПлюс.

27. Власов А. А.Метод контурной сегментации канни с использованием морфологических операций // Решетневские чтения. – 2010. – № 2. – С. 479.

ПРИЛОЖЕНИЕ А

Техническое задание

СОГЛАСОВАНО	УТВЕРЖДАЮ
доцент кафедры ОД, к.т.н. - ПНИПУ	Доцент с и.о. зав. Кафедрой ОНД
_____ А.А. Петренко	_____ Е.Н. Хаматнурова
«___»_____2020	«___»_____2020

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПОЛУЧЕНИЯ РАЗМЕРОВ ДЕЛОВОГО ОСТАТКА НЕСТАНДАРТНОЙ ФОРМЫ, ПО ЦИФРОВОЙ ФОТОГРАФИИ

На 16 листах
Действует с 01.10.2019

Разработала студентка гр. ЭВТ-16-1б
_____ В.Р. Зайникова
«___»_____2020

СОДЕРЖАНИЕ

1 Общие положения

Работы по разработке информационной системы производятся в соответствии с документами:

– техническое задание на разработку программного обеспечения получения размеров делового остатка нестандартной формы, по цифровой фотографии;

– учебный план специальности «Вычислительные, машины, комплексы, системы и сети».

1.1 Полное наименование системы и ее условное обозначение

Настоящая выпускная квалификационная работа определяет требования порядок Разработка программного обеспечения получения размеров делового остатка нестандартной формы, по цифровой фотографии

Условное обозначение системы – «ПОПРДОНФЦФ».

Краткое наименование системы: ПО получения размеров делового остатка

1.2 Основания для разработки

Работа выполняется на основании учебного плана направления «09.03.01 – Информатика и вычислительная техника», профиль: «Вычислительные машины, комплексы, системы и сети» и темы выпускной квалификационной работы, согласованной с заведующим кафедрой ОНД ЛФ ПНИПУ, Хаматнуровой Е.Н..

1.3 Наименования организации-заказчика и организаций - участников работ

Заказчик: ОНД ЛФ ПНИПУ.

Адрес фактический: г. Лысьва, ул. Ленина 2.

Телефон: 61255

Разработчик:

Студентка группы ЭВТ-16-1б Зайникова Виктория Рустемовна

Телефон: +7(952)327-69-19.

1.4 Плановые сроки начала и окончания работы по созданию системы

Срок начала разработки ПО получения размеров делового остатка 01.10.19.

Срок разработки ПО получения размеров делового остатка регламентирован действующим учебным планом и предполагает сдачу проекта системы 25.06.20.

1.5 Источники и порядок финансирования работ

Финансирование работ не предусмотрено.

1.6 Порядок оформления и предъявления заказчику результатов работ по созданию системы

Работы по разработке системы проводятся и принимаются поэтапно. После окончания каждого этапа, установленных учебным планом, студент предоставляет результаты работ руководителю выпускной квалификационной работы. Руководитель оценивает работу.

1.7. Перечень нормативно-технических документов, методических материалов, использованных при разработке ТЗ

При разработке технического задания на разработку программного обеспечения получения размеров делового остатка нестандартной формы, по цифровой фотографии, исполнитель должен руководствоваться требованиями следующих нормативных документов:

– ГОСТ 34.601-90 – Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания;

– ГОСТ 34.201-89 – Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплексность и обозначение документов при создании автоматизированных систем.

2 Назначение и цели создания системы

2.1 Назначение системы

Основным назначением является автоматизация процесса получения размеров ДО.

2.2 Цели создания системы

Целью создания заключается в проектировании и создании программного обеспечения получения размеров ДО нестандартной формы, по цифровой фотографии.

Для реализации поставленной цели сформулированы следующие задачи:

1) исследовать модели представления предметной области;

2) описать предметную область в виде определённой модели представления знаний;

3) рассмотреть библиотеки для реализации ПО;

4) разработать систему получения размеров ДО.

3 Требования к системе

Разрабатываемая система должна соответствовать требованиям технического задания на ее создание.

Автоматизированная система в целом и все виды ее обеспечения должны быть приспособлены к модернизации, развитию и наращиванию. Все программные средства должны быть лицензированными.

3.1 Требования к системе в целом

3.1.1 Требования к структуре и функционированию системы

3.1.1.1 Перечень подсистем, их назначение и основные характеристики, требования к числу уровней иерархии и степени централизации системы

Разрабатываемая система должна быть централизованной. Все ее данные должны располагаться в центральном хранилище.

3.1.1.2 Требования к способам и средствам связи для информационного обмена между компонентами системы

Все компоненты системы должны функционировать в пределах единого логического пространства.

3.1.1.3 Требования к характеристикам взаимосвязей создаваемой системы со смежными системами

Программное обеспечение системы должно обеспечивать интеграцию и совместимость на информационном уровне с другими системами.

3.1.1.4 Требования к режимам функционирования системы

Система должна выполнять свои функции непрерывно (круглосуточно). Работа пользователей возможна 24 часа в день, 7 дней в неделю.

3.1.1.5 Перспективы развития, модернизации системы

Система должна разрабатываться с учетом обеспечения ее дальнейшего развития и наращивания функциональности.

3.1.2 Требования к численности и квалификации персонала системы и режиму его работы

Для поддержки функционирования автоматизированной системы заказчиком должна быть создана служба эксплуатации, персонал, который должен обладать знаниями в области информационных платформ, на которых будет реализована система.

В состав персонала, необходимого для обеспечения эксплуатации комплекса средств автоматизации системы, должны входить:

1. Администратор системы;
2. Пользователь системы;
3. Программист.

Администратор системы должен обладать высокой квалификацией и опытом выполнения работ по внедрению, настройке и администрированию программных средств, применяемых в системе.

Пользователь системы должен обладать знаниями в соответствующей предметной области, иметь опыт работы с персональным компьютером. В обязанности пользователя входит запуск интерфейса системы, выбор необходимых объектов, ввод текстовой информации в соответствующие поля. Пользователь должен следить за корректностью входной и выходной информации.

В обязанности программиста входит модернизация, диагностика и устранение выявленных ошибок в системе.

Режим работы персонала устанавливается в соответствии с основным рабочим графиком подразделений Заказчика.

3.1.3 Требования к надежности

При возникновении сбоев в аппаратном обеспечении, включая аварийное отключение электропитания, разрабатываемая система должна автоматически восстанавливать свою работоспособность после устранения сбоев и корректного перезапуска аппаратного обеспечения (за исключением случаев повреждения рабочих носителей информации с исполняемым программным кодом).

Система должна обеспечивать корректную обработку аварийных ситуаций, вызванных неверными действиями пользователей, неверным форматом или недопустимыми значениями входных данных. В указанных случаях пользователю должны выдаваться соответствующие аварийные сообщения, после чего возвращаться в рабочее состояние, предшествовавшее неверной (недопустимой) команде, группы команд или некорректному вводу данных.

3.1.4 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы

Техническое обслуживание системы должен производить высококвалифицированный сотрудник с соответствующим образованием. Обслуживание системы низко квалифицированными сотрудниками строго нежелательно. Ремонт системы выполняется исключительно при полностью выключенной системе.

3.1.5 Требования к защите информации от несанкционированного доступа

Система должна обеспечивать защиту от несанкционированного доступа (НСД) без привязки к нормативам, предъявляемым к категории 1Г по классификации действующего руководящего документа ФСТЭК России «Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем».

Требования к антивирусной защите:

Средства антивирусной защиты должны быть установлены на всех рабочих местах пользователей и администраторов системы.

Средства антивирусной защиты рабочих мест пользователей и администраторов должны обеспечивать:

- централизованное управление сканированием, удалением вирусов и протоколированием вирусной активности на рабочих местах пользователей;

- централизованное автоматическое обновление вирусных сигнатур на рабочих местах пользователей и администраторов;

- ведение журналов вирусной активности;

- администрирование всех антивирусных продуктов.

3.1.6 Требования по сохранности информации при авариях

При авариях сохранность информации должна обеспечиваться, питанием ЭВМ и серверов от бесперебойного источника питания. Программное обеспечение информационной системы должно автоматически восстанавливать свое функционирование после аварии при корректном перезапуске аппаратных средств. Должна быть предусмотрена возможность организации автоматического или ручного резервного копирования данных.

Перечень аварийных ситуаций, при которых должна быть обеспечена сохранность информации:

- отключение питания;

- ошибки в системе, при которых произошло аварийное завершение работы;

- программа не отвечает на действия пользователей;

- потеря содержимого оперативной памяти (мягкий сбой);

- поломка основного носителя базы данных (жесткий сбой).

3.1.7 Требования к защите от влияния внешних воздействий

При работе с данным компонентом допускаются только сотрудники, прошедшие соответствующие курсы.

3.1.8 Требования к патентной чистоте

Программное и аппаратное обеспечение, используемое при реализации проекта, должно иметь соответствующие лицензии на его использование, быть сертифицировано для работы в используемых режимах.

3.1.9 Требования по стандартизации и унификации

Требования унификации при проектировании интерфейса.

- В разделах интерфейса для обозначения сходных операций должны использоваться сходные графические значки, кнопки и т.п. управляющие (навигационные) элементы.

- Термины, используемые для обозначения типовых операций, а также последовательности действий пользователя при их выполнении, должны быть унифицированы.

- Внешнее поведение сходных элементов интерфейса (реакция на наведение указателя «мыши», переключение фокуса, нажатие кнопки и т.п.) должны реализовываться одинаково для однотипных элементов.

3.1.10 Дополнительные требования

Требования не предъявляются.

3.2 Требования к функциям (задачам), выполняемым системой

Разрабатываемое ПО должно обладать следующими функциями:

1. Загрузка фотографии с изображением ДО;
2. Исправление перспективы для получения изображениявида «сверху - вниз»;

3. Автоматическое выделение контура с возможностью его редактирования;

4. Расчет площади выделенного контура;

5. Ввод данных контура в базу данных.

3.3 Требования к видам обеспечения

3.3.1 Требования к математическому обеспечению

Алгоритмы, используемые в системе, должны быть разработаны с учетом возможности получения некорректной входной информации и предусматривать соответствующую реакцию на такие события.

3.3.2 Требования к информационному обеспечению

Информационное обеспечение целесообразно строить на основе системы реляционных баз данных.

3.3.3 Требования к лингвистическому обеспечению

Все интерфейсы должны быть выполнены на русском языке.

Все инструкции и руководства для пользователей и обслуживающего персонала системы должны быть разработаны на русском языке.

Система должна поддерживать Unicode-представления данных для хранения и отображения, данных хранилища на разных языках.

3.3.4 Требования к программному обеспечению

Системное программное обеспечение рабочих станций должно удовлетворять следующим условия:

- Операционная система MS Windows 7/8/10 Корпоративная с установленной графической оболочкой;

- ПО должно быть разработано с помощью языка python, с использованием библиотек прогрограммных модулей

3.3.5 Требования к техническому обеспечению

Требования к оборудованию:

- Процессор – IntelCore i3 и выше;

- Оперативная память – 4GB;

Система должна гарантированно функционировать на технических средствах и программно-аппаратных платформах, используемых в отделах компании-заказчика.

3.3.6 Требования к организационному обеспечению

Организационное обеспечение системы должно быть достаточным для эффективного выполнения персоналом возложенных на него обязанностей при осуществлении автоматизированных и связанных с ними неавтоматизированных функций системы.

К работе с системой должны допускаться сотрудники, имеющие навыки работы на персональном компьютере, ознакомленные с правилами эксплуатации и прошедшие обучение работе с системой.

4. Состав и содержание работ по созданию системы

В соответствии с ГОСТ 34.601-90 работы по созданию системы генерации моделей на основе рентгеновских изображений выполняются с учетом этапов, приведенных в таблице 1.

Таблица В.1

	Содержание этапа	Срок	Результат
	Обследование объекта автоматизации	09.01.20 – 09.02.20	Отчет о результатах исследования
	Разработка технического задания на создание системы	10.02.20 – 10.03.2020	Утверждение заказчиком ТЗ
	Разработка, откладка и тестирование программных средств	11.03.20 – 31.05.20	Выполняемые программные модули
	Проведение испытаний	31.05.20 – 07.06.20	Протокол испытаний

5 Порядок контроля и приемки системы

5.1 Виды и объем испытаний системы

Для ПО получения размеров ДО установлены следующие виды испытаний:

Опытная эксплуатация проводится на тестовых данных специалистом со стороны Заказчика. В этом случае клиент предоставляет необходимые тестовые данные. На данном этапе в систему денотативного анализа заносятся тестовые данные, по результатам тестирования формируются отчетные документы.

5.2 Общие требования к приемке работ по стадиям

Все работы по приемке ПО получения размеров ДО осуществляются комиссией, состоящей из представителей Заказчика и Исполнителя.

Программное изделие, создаваемое в рамках представленной работы, передается Заказчику в виде готовых модулей, предоставляемых в электронном формате на стандартном машинном носителе.

6 Требования к составу и содержанию работа по подготовке объекта автоматизации к вводу системы в действие

Мероприятия по вводу программных средств системы в эксплуатацию включают в себя следующие работы:

-установка ПО на оборудовании Заказчика.

-обучение пользователей порядку работы с соответствующим функционалом в течение всего периода опытной эксплуатации.

6.1 Требования к документированию

Пакет программной документации на разрабатываемую систему должны входить документы, представленные в таблице В.2.

Таблица В.2

Стадия (этап) создания системы	Предъявляемые документы
Разработка проектных решений по системе и ее частям	Техническое задание
Разработка рабочей документации на систему и ее части	Документация «Руководство пользователя»
Внедрение	Акт приемки в опытную эксплуатацию

Вся документация должна быть подготовлена в напечатанном виде, интегрируемыми со средствами проектирования и разработки прикладного программного обеспечения. Состав документации может быть уточнен в ходе проектирования. Настоящее техническое задание может дополняться и изменяться в процессе разработки и испытаний в установленном порядке по взаимному соглашению Заказчика и Исполнителя.

СОСТАВИЛИ

Наименование организации, предприятия	Должность исполнителя	Фамилия имя, отчество	Подпись	Дата
ЛФ ПНИПУ	студент	Зайникова Виктория Рустемовна		

СОГЛАСОВАНО

Наименование организации, предприятия	Должность исполнителя	Фамилия имя, отчество	Подпись	Дата
ЛФ ПНИПУ	Кандидат технических наук	Александр Анатольевич Петренко		

ПРИЛОЖЕНИЕ Б

Руководство пользователя

Прилагается диск с руководством пользователя в формате pdf.

ПРИЛОЖЕНИЕ В

Листинг форм ПО

```
# подключаем библиотеки
import os, shutil
import sys
import cv2
import numpy as np
import math
from PIL import Image
from scipy.spatial import distance as dist
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
from matplotlib.artist import Artist
from matplotlib.patches import Polygon
from comtypes.client import CreateObject
from comtypes.gen import Access
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QApplication, QMainWindow,  QMessageBox
# путь к файлу
F=""
F1=""
n_points=[]
#=====================================================================#
класс для выделения области преобразования перспективы
class PolygonInteractor(object):
    showverts = True
    # максимальное расстояние в пикселях для подсчета попадания в вершину
    epsilon = 5
    def __init__(self, ax, poly):
        self.ax = ax
        canvas = poly.figure.canvas
        self.poly = poly

        x, y = zip(*self.poly.xy)
        self.line = Line2D(x, y, marker='o', markerfacecolor='r',
animated=True)
        self.ax.add_line(self.line)

        cid = self.poly.add_callback(self.poly_changed)
        self._ind = None

        canvas.mpl_connect('draw_event', self.draw_callback)
        canvas.mpl_connect('button_press_event',
self.button_press_callback)
        canvas.mpl_connect('button_release_event',
self.button_release_callback)
        canvas.mpl_connect('motion_notify_event',
self.motion_notify_callback)
        self.canvas = canvas

    def get_poly_points(self):
        return np.asarray(self.poly.xy)

    def draw_callback(self, event):
        self.background = self.canvas.copy_from_bbox(self.ax.bbox)
```

```
        self.ax.draw_artist(self.poly)
        self.ax.draw_artist(self.line)
        self.canvas.blit(self.ax.bbox)

    def poly_changed(self, poly):
        vis = self.line.get_visible()
        Artist.update_from(self.line, poly)
        self.line.set_visible(vis)

    def get_ind_under_point(self, event):
        xy = np.asarray(self.poly.xy)
        xyt = self.poly.get_transform().transform(xy)
        xt, yt = xyt[:, 0], xyt[:, 1]
        d = np.sqrt((xt - event.x)**2 + (yt - event.y)**2)
        indseq = np.nonzero(np.equal(d, np.amin(d)))[0]
        ind = indseq[0]

        if d[ind] >= self.epsilon:
            ind = None
        return ind

    def button_press_callback(self, event):
        if not self.showverts:
            return
        if event.inaxes is None:
            return
        if event.button != 1:
            return
        self._ind = self.get_ind_under_point(event)

    def button_release_callback(self, event):
        if not self.showverts:
            return
        if event.button != 1:
            return
        self._ind = None

    def motion_notify_callback(self, event):
        if not self.showverts:
            return
        if self._ind is None:
            return
        if event.inaxes is None:
            return
        if event.button != 1:
            return
        x, y = event.xdata, event.ydata

        self.poly.xy[self._ind] = x, y
        if self._ind == 0:
            self.poly.xy[-1] = x, y
        elif self._ind == len(self.poly.xy) - 1:
            self.poly.xy[0] = x, y
        self.line.set_data(zip(*self.poly.xy))

        self.canvas.restore_region(self.background)
        self.ax.draw_artist(self.poly)
        self.ax.draw_artist(self.line)
```

```
        self.canvas.blit(self.ax.bbox)
#======================================================================
# класс для коррекции перспективы
class perspectiva(object):
    # выделяем область для коррекции изображения
    def get_contour(self, rescaled_image):
        IM_HEIGHT, IM_WIDTH, _ = rescaled_image.shape
        TOP_RIGHT = (IM_WIDTH - 50, 50)
        BOTTOM_RIGHT = (IM_WIDTH - 50, IM_HEIGHT - 50)
        BOTTOM_LEFT = (50, IM_HEIGHT - 50)
        TOP_LEFT = (50, 50)
        screenCnt = np.array([[TOP_RIGHT], [BOTTOM_RIGHT], [BOTTOM_LEFT],
[TOP_LEFT]])
        return screenCnt.reshape(4, 2)

    def interactive_get_contour(self, screenCnt, open_image):
        poly = Polygon(screenCnt, animated=True,  fill=False,
color="blue")
        fig, ax = plt.subplots()
        ax.add_patch(poly)
        ax.set_title(('Выделите область для коррекции перспективы делового
остатка. \n'
                      'Закройте окно, когда закончите.'))
        p = PolygonInteractor(ax, poly)
        plt.imshow(open_image)
        plt.show()
        new_points = p.get_poly_points()[:4]
        new_points = np.array([[p] for p in new_points], dtype="int32")
        return new_points.reshape(4, 2)

    def resize(self,image, width=None, height=None, inter=cv2.INTER_AREA):
        # инициализировать размеры изображения, которое нужно изменить, и
захватить размер изображения
        dim = None
        (h, w) = image.shape[:2]

        # если ширина и высота None, вернуть исходное изображение
        if width is None and height is None:
            return image

        # проверить, если ширина равна None
        if width is None:
            # рассчитать соотношение высоты и построить размеры
            r = height / float(h)
            dim = (int(w * r), height)

        # в противном случае высота равна None
        else:
            # рассчитать соотношение ширины и построить размеры
            r = width / float(w)
            dim = (width, int(h * r))

        # изменить размер изображения
        resized = cv2.resize(image, dim, interpolation=inter)

        # return изображение
        return resized
```

```
    def order_points(self, pts):
        # сортировать точки по их координатам x
        xSorted = pts[np.argsort(pts[:, 0]), :]

        # захватить самые левые и самые правые точки из отсортированных x
        leftMost = xSorted[:2, :]
        rightMost = xSorted[2:, :]

        # теперь сортируем крайние левые координаты по их y-координатам,
        # чтобы мы могли получить верхнюю левую и нижнюю левую точки
соответственно
        leftMost = leftMost[np.argsort(leftMost[:, 1]), :]
        (tl, bl) = leftMost

        # теперь, когда у нас есть верхняя левая координата,
        # используйте ее в качестве якоря для вычисления евклидова
расстояния между верхней левой и самой правой точками;
        # по теореме Пифагора, точка с наибольшим расстоянием будет нашей
правой нижней точкой
        D = dist.cdist(tl[np.newaxis], rightMost, "euclidean")[0]
        (br, tr) = rightMost[np.argsort(D)[::-1], :]

        # вернуть координаты в верхней левой, верхней правой, нижний
правый и нижний левый точек
        return np.array([tl, tr, br, bl], dtype="float32")

    def four_point_transform(self, image, pts):
        # получить последовательный порядок точек и распаковать их
индивидуально
        rect = self.order_points(pts)
        (tl, tr, br, bl) = rect

        # вычислить ширину нового изображения,
        # которая будет максимальным расстоянием между нижними правыми и
нижними левыми x-координатами
        # или верхними правыми и верхними левыми x-координатами
        widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
        widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
        maxWidth = max(int(widthA), int(widthB))

        # вычислить высоту нового изображения,
        # которая будет максимальным расстоянием между верхними правыми и
нижними правыми y-координатами
        # или верхними левыми и нижними левыми y-координатами
        heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
        heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
        maxHeight = max(int(heightA), int(heightB))

        # есть размеры нового изображения,
        # создайли набор точек назначения,
        # чтобы получить 90 градусов
        dst = np.array([
            [0, 0],
            [maxWidth - 1, 0],
            [maxWidth - 1, maxHeight - 1],
            [0, maxHeight - 1]], dtype="float32")

        # вычислить  perspective transform matrix
```

```
        M = cv2.getPerspectiveTransform(rect, dst)
        warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))

        return warped

    def obrabotka(self, image_open):
        RESCALED_HEIGHT = 500.0
        # загрузить изображение и вычислить отношение старой высоты к
новой высоте, клонировать и изменить его размер
        image = cv2.imread(image_open)
        assert (image is not None)
        ratio = image.shape[0] / RESCALED_HEIGHT
        orig = image.copy()
        rescaled_image = self.resize(image, height=int(RESCALED_HEIGHT))
        # получить контур
        screenCnt = self.get_contour(rescaled_image)
        screenCnt = self.interactive_get_contour(screenCnt,
rescaled_image)
        # применить коррекцию перспективы
        warped = self.four_point_transform(orig, screenCnt * ratio)
        # сохранить преобразованное изображение
        cv2.imwrite('temp/transform_img.jpg', warped)

#================================================================#
главное окно
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(267, 308)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("image/tema.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        MainWindow.setWindowIcon(icon)
        MainWindow.setAutoFillBackground(False)
        MainWindow.setToolButtonStyle(QtCore.Qt.ToolButtonIconOnly)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.pushButton_2 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_2.setGeometry(QtCore.QRect(20, 10, 231, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.pushButton_2.setFont(font)
        icon1 = QtGui.QIcon()
        icon1.addPixmap(QtGui.QPixmap("image/progekt.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.pushButton_2.setIcon(icon1)
        self.pushButton_2.setIconSize(QtCore.QSize(24, 24))
        self.pushButton_2.setObjectName("pushButton_2")
        self.pushButton_3 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_3.setGeometry(QtCore.QRect(20, 110, 231, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.pushButton_3.setFont(font)
        icon2 = QtGui.QIcon()
        icon2.addPixmap(QtGui.QPixmap("image/spravca.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.pushButton_3.setIcon(icon2)
        self.pushButton_3.setIconSize(QtCore.QSize(24, 24))
```

```
        self.pushButton_3.setObjectName("pushButton_3")
        self.pushButton_4 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_4.setGeometry(QtCore.QRect(20, 160, 231, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.pushButton_4.setFont(font)
        icon3 = QtGui.QIcon()
        icon3.addPixmap(QtGui.QPixmap("image/o_prooect.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.pushButton_4.setIcon(icon3)
        self.pushButton_4.setIconSize(QtCore.QSize(24, 24))
        self.pushButton_4.setObjectName("pushButton_4")
        self.pushButton_5 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_5.setGeometry(QtCore.QRect(20, 210, 231, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.pushButton_5.setFont(font)
        self.pushButton_5.setIconSize(QtCore.QSize(24, 24))
        self.pushButton_5.setObjectName("pushButton_5")
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(20, 60, 231, 51))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.pushButton.setFont(font)
        icon4 = QtGui.QIcon()
        icon4.addPixmap(QtGui.QPixmap("image/progekt_R.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.pushButton.setIcon(icon4)
        self.pushButton.setIconSize(QtCore.QSize(24, 24))
        self.pushButton.setObjectName("pushButton")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 267, 21))
        self.menubar.setObjectName("menubar")
        self.menu = QtWidgets.QMenu(self.menubar)
        self.menu.setObjectName("menu")
        self.menu_2 = QtWidgets.QMenu(self.menubar)
        self.menu_2.setObjectName("menu_2")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.action = QtWidgets.QAction(MainWindow)
        self.action.setIcon(icon1)
        self.action.setObjectName("action")
        self.action_2 = QtWidgets.QAction(MainWindow)
        self.action_2.setObjectName("action_2")
        self.action_3 = QtWidgets.QAction(MainWindow)
        self.action_3.setIcon(icon2)
        self.action_3.setObjectName("action_3")
        self.action_4 = QtWidgets.QAction(MainWindow)
        self.action_4.setIcon(icon3)
        self.action_4.setObjectName("action_4")
        self.action_5 = QtWidgets.QAction(MainWindow)
        self.action_5.setIcon(icon4)
        self.action_5.setObjectName("action_5")
        self.menu.addAction(self.action)
        self.menu.addAction(self.action_5)
```

```
        self.menu.addAction(self.action_2)
        self.menu_2.addAction(self.action_3)
        self.menu_2.addAction(self.action_4)
        self.menubar.addAction(self.menu.menuAction())
        self.menubar.addAction(self.menu_2.menuAction())

        self.pushButton_5.clicked.connect(self.close)
        self.pushButton_3.clicked.connect(self.show_spravka)
        self.action_2.triggered.connect(self.close)
        self.action_3.triggered.connect(self.show_spravka)
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Главное
окно"))
        self.pushButton_2.setText(_translate("MainWindow", "Создать
проект"))
        self.pushButton_3.setText(_translate("MainWindow", " Справка"))
        self.pushButton_4.setText(_translate("MainWindow", "О программе"))
        self.pushButton_5.setText(_translate("MainWindow", " Выход"))
        self.pushButton.setText(_translate("MainWindow", "Редактировать
проект"))
        self.menu.setTitle(_translate("MainWindow", "Проект"))
        self.menu_2.setTitle(_translate("MainWindow", "Помощь"))
        self.action.setText(_translate("MainWindow", "Создать проект"))
        self.action_2.setText(_translate("MainWindow", "Выход"))
        self.action_3.setText(_translate("MainWindow", "Справка"))
        self.action_4.setText(_translate("MainWindow", "О программе"))
        self.action_5.setText(_translate("MainWindow", "Редактировать
проект"))

    def show_spravka(self):
        os.startfile(r'User_manual.pdf')

# ================================================================================
# проект
class progect(object):
   def setupUi(self, MainWindow):
       MainWindow.setObjectName("MainWindow")
       MainWindow.resize(660, 490)
       icon = QtGui.QIcon()
       icon.addPixmap(QtGui.QPixmap("image/tema.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
       MainWindow.setWindowIcon(icon)
       self.centralwidget = QtWidgets.QWidget(MainWindow)
       self.centralwidget.setObjectName("centralwidget")
       self.image = QtWidgets.QLabel(self.centralwidget)
       self.image.setGeometry(QtCore.QRect(220, 20, 411, 371))
       self.image.setTabletTracking(False)
       self.image.setFrameShape(QtWidgets.QFrame.Box)
       self.image.setText("")
       self.image.setPixmap(QtGui.QPixmap("image/no_image.png"))
       self.image.setScaledContents(True)
       self.image.setObjectName("image")
       self.perspektiva = QtWidgets.QPushButton(self.centralwidget)
       self.perspektiva.setGeometry(QtCore.QRect(10, 80, 201, 61))
```

```
        font = QtGui.QFont()
        font.setPointSize(12)
        self.perspektiva.setFont(font)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("image/persp.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.perspektiva.setIcon(icon)
        self.perspektiva.setIconSize(QtCore.QSize(20, 20))
        self.perspektiva.setObjectName("perspektiva")
        self.perspektiva.setEnabled(False)
        self.raschet = QtWidgets.QPushButton(self.centralwidget)
        self.raschet.setGeometry(QtCore.QRect(10, 200, 201, 61))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.raschet.setFont(font)
        icon1 = QtGui.QIcon()
        icon1.addPixmap(QtGui.QPixmap("image/area.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.raschet.setIcon(icon1)
        self.raschet.setIconSize(QtCore.QSize(20, 20))
        self.raschet.setObjectName("raschet")
        self.raschet.setEnabled(False)
        self.contur = QtWidgets.QPushButton(self.centralwidget)
        self.contur.setGeometry(QtCore.QRect(10, 140, 201, 61))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.contur.setFont(font)
        icon2 = QtGui.QIcon()
        icon2.addPixmap(QtGui.QPixmap("image/contur.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.contur.setIcon(icon2)
        self.contur.setIconSize(QtCore.QSize(20, 20))
        self.contur.setObjectName("contur")
        self.contur.setEnabled(False)
        self.conttable = QtWidgets.QPushButton(self.centralwidget)
        self.conttable.setGeometry(QtCore.QRect(10, 260, 201, 61))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.conttable.setFont(font)
        icon3 = QtGui.QIcon()
        icon3.addPixmap(QtGui.QPixmap("image/database.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.conttable.setIcon(icon3)
        self.conttable.setIconSize(QtCore.QSize(20, 20))
        self.conttable.setObjectName("conttable")
        self.conttable.setEnabled(False)
        self.zakryt = QtWidgets.QPushButton(self.centralwidget)
        self.zakryt.setGeometry(QtCore.QRect(530, 400, 101, 41))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.zakryt.setFont(font)
        icon4 = QtGui.QIcon()
        icon4.addPixmap(QtGui.QPixmap("image/close.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.zakryt.setIcon(icon4)
        self.zakryt.setIconSize(QtCore.QSize(16, 16))
        self.zakryt.setObjectName("zakryt")
        self.saveprogekt = QtWidgets.QPushButton(self.centralwidget)
```

```
        self.saveprogekt.setGeometry(QtCore.QRect(400, 400, 121, 41))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.saveprogekt.setFont(font)
        icon5 = QtGui.QIcon()
        icon5.addPixmap(QtGui.QPixmap("image/save.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.saveprogekt.setIcon(icon5)
        self.saveprogekt.setIconSize(QtCore.QSize(20, 20))
        self.saveprogekt.setObjectName("saveprogekt")
        self.saveprogekt.setEnabled(False)
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(10, 20, 201, 61))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.pushButton.setFont(font)
        icon6 = QtGui.QIcon()
        icon6.addPixmap(QtGui.QPixmap("image/loag.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.pushButton.setIcon(icon6)
        self.pushButton.setIconSize(QtCore.QSize(24, 24))
        self.pushButton.setObjectName("pushButton")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 646, 21))
        self.menubar.setObjectName("menubar")
        self.menu = QtWidgets.QMenu(self.menubar)
        self.menu.setObjectName("menu")
        self.menu_2 = QtWidgets.QMenu(self.menubar)
        self.menu_2.setObjectName("menu_2")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.action = QtWidgets.QAction(MainWindow)
        self.action.setIcon(icon5)
        self.action.setObjectName("action")
        self.action.setEnabled(False)
        self.action_2 = QtWidgets.QAction(MainWindow)
        self.action_2.setIcon(icon4)
        self.action_2.setObjectName("action_2")
        self.action_3 = QtWidgets.QAction(MainWindow)
        icon7 = QtGui.QIcon()
        icon7.addPixmap(QtGui.QPixmap("image/spravca.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.action_3.setIcon(icon7)
        self.action_3.setObjectName("action_3")
        self.action_4 = QtWidgets.QAction(MainWindow)
        icon8 = QtGui.QIcon()
        icon8.addPixmap(QtGui.QPixmap("image/o_prooect.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.action_4.setIcon(icon8)
        self.action_4.setObjectName("action_4")
        self.menu.addAction(self.action)
        self.menu.addAction(self.action_2)
        self.menu_2.addAction(self.action_3)
        self.menu_2.addAction(self.action_4)
        self.menubar.addAction(self.menu.menuAction())
```

```
        self.menubar.addAction(self.menu_2.menuAction())

        self.pushButton.clicked.connect(self.view_img)
        self.perspektiva.clicked.connect(self.persp_view)
        self.zakryt.clicked.connect(self.close)
        self.zakryt.clicked.connect(self.close)
        self.raschet.clicked.connect(self.find_cntr)
        self.contur.clicked.connect(self.cntr_img)
        self.conttable.clicked.connect(self.open_database)
        self.saveprogekt.clicked.connect(self.save_project)
        self.action.triggered.connect(self.save_project)
        self.action_2.triggered.connect(self.close)
        self.action_3.triggered.connect(self.show_spravka)
        self.action_4.triggered.connect(self.show_window_4)
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Проект"))
        self.perspektiva.setToolTip(
            _translate("MainWindow", "<html><head/><body><p>Исправить
изображение</p></body></html>"))
        self.perspektiva.setText(_translate("MainWindow", " Перспектива"))
        self.raschet.setToolTip(_translate("MainWindow",
"<html><head/><body><p>Площадь детали</p></body></html>"))
        self.raschet.setWhatsThis(
            _translate("MainWindow", "<html><head/><body><p>Рассчитать
площадь детали</p></body></html>"))
        self.raschet.setText(_translate("MainWindow", " Расчет"))
        self.contur.setToolTip(
            _translate("MainWindow", "<html><head/><body><p>Вывести контур
детали</p></body></html>"))
        self.contur.setText(_translate("MainWindow", " Контур"))
        self.conttable.setToolTip(
            _translate("MainWindow", "<html><head/><body><p>Открыть базу
данных с координатами</p></body></html>"))
        self.conttable.setText(_translate("MainWindow", "Координаты"))
        self.zakryt.setText(_translate("MainWindow", "Закрыть"))
        self.saveprogekt.setText(_translate("MainWindow", "Сохранить"))
        self.pushButton.setText(_translate("MainWindow", "  Загрузить"))
        self.menu.setTitle(_translate("MainWindow", "Проект"))
        self.menu_2.setTitle(_translate("MainWindow", "Помощь"))
        self.action.setText(_translate("MainWindow", "Сохранить"))
        self.action_2.setText(_translate("MainWindow", "Выход"))
        self.action_3.setText(_translate("MainWindow", "Справка"))
        self.action_4.setText(_translate("MainWindow", "О программе"))

    def show_spravka(self):
        os.startfile(r'User_manual.pdf')

    # загружаем изображение
    def loag_img(self):
        fileName = QtWidgets.QFileDialog.getOpenFileName(self,
                                                         "Выберите файл",
QtCore.QDir.homePath(),
                                                         "Изображение
(*.jpg)")[0]
```

```
        return fileName

    def view_img(self):
        fileName = self.loag_img()
        if fileName == "":
            QtWidgets.QMessageBox.information(self, "Ошибка",
                                              "Не удалось открыть
изображение")
        else:
            img = cv2.imread(fileName)
            cv2.imwrite('temp/open_img.jpg', img)
            self.image.setPixmap(QtGui.QPixmap("temp/open_img.jpg"))
            self.perspektiva.setEnabled(True)
            self.contur.setEnabled(False)
            self.raschet.setEnabled(False)
            self.conttable.setEnabled(False)
            self.saveprogekt.setEnabled(False)
            self.action.setEnabled(False)
            pass

    # исправляем перспективу
    def correction_img(self):
         image = 'temp/open_img.jpg'
         pers = perspectiva()
         pers.obrabotka(image)
         self.image.setPixmap(QtGui.QPixmap("temp/transform_img.jpg"))

    def persp_view(self):
        # делаем кнопку неактивной
        self.correction_img()
        self.perspektiva.setEnabled(True)
        self.contur.setEnabled(True)
        self.raschet.setEnabled(False)
        self.conttable.setEnabled(False)
        self.saveprogekt.setEnabled(False)
        self.action.setEnabled(False)
        pass

    #ввывод контура
    def cntr_img(self):
        image = cv2.imread('temp/transform_img.jpg')
        image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        image_edge = cv2.Canny(image_gray, 80, 160)

        cnts, hiers = cv2.findContours(image_edge.copy(), cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)[-2:]
        cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:5]
        screenCnt = []
        for c in cnts:
            peri = cv2.arcLength(c, True)
            approx = cv2.approxPolyDP(c, 0.02 * peri, True)
            if len(approx) == 4:
                screenCnt = approx
                break
        # площадь листа А4 62370  А4 - 210х297

        if len(screenCnt) == 4:
            xy1 = screenCnt[0][0]
```

```
            xy2 = screenCnt[1][0]
            xy3 = screenCnt[2][0]
            xy4 = screenCnt[3][0]
            x1 = xy1[0]
            y1 = xy1[1]
            x2 = xy2[0]
            y2 = xy2[1]
            x3 = xy3[0]
            y3 = xy3[1]
            x4 = xy4[0]
            y4 = xy4[1]

            d1 = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
            d2 = math.sqrt((x3 - x2) ** 2 + (y3 - y2) ** 2)
            d3 = math.sqrt((x4 - x3) ** 2 + (y4 - y3) ** 2)
            d4 = math.sqrt((x1 - x4) ** 2 + (y1 - y4) ** 2)

            if d1 > d2:
                width = (d2 + d4) / 2
                height = (d1 + d3) / 2
            else:
                width = (d1 + d3) / 2
                height = (d2 + d4) / 2

            w = width
            h = height

            image_R = Image.open('temp/transform_img.jpg')
            width_i, height_i = image_R.size
            width_A4 = 210
            height_A4 = 297
            if h > height_A4:
                h_i = math.ceil(height_i / (h / height_A4))
            else:
                h_i = math.ceil(height_i * (height_A4 / h))
            if w > width_A4:
                w_i = math.ceil(width_i / (w / width_A4))
            else:
                w_i = math.ceil(width_i * (width_A4 / w))
            size_R = (h_i, w_i)
            image_Res = image_R.resize(size_R)
            image_Res.save('temp/size_img.jpg')

            self.image.setPixmap(QtGui.QPixmap("temp/size_img.jpg"))

            image = cv2.imread('temp/size_img.jpg')
            image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            image_edge = cv2.Canny(image_gray, 80, 160)
            cnts, hiers = cv2.findContours(image_edge.copy(),
cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)[-2:]
            cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:5]
            for c in cnts:
                peri = cv2.arcLength(c, True)
                approx = cv2.approxPolyDP(c, 0.02 * peri, True)
                if len(approx) == 4:
                    screenCnt = approx
                    break
            sq = cv2.contourArea(screenCnt)
```

```
            if ((sq >= 65000) or (sq <= 61000)):
                msgBox = QMessageBox()
                msgBox.setIcon(QMessageBox.Information)
                msgBox.setText("Контур найден некорректно! Исправте
перспективу или загрузите другое изображение")
                msgBox.setWindowTitle("Ошибка")
                returnValue = msgBox.exec()
            else:
                image = cv2.imread("temp/size_img.jpg")
                edged = cv2.Canny(image, 10, 250)
                kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7))
                closed = cv2.morphologyEx(edged, cv2.MORPH_CLOSE, kernel)
                sharpen = cv2.GaussianBlur(closed, (0, 0), 3)
                sharpen = cv2.addWeighted(closed, 1.5, sharpen, -0.5, 0)
                cnts, _ = cv2.findContours(sharpen, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
                # найдем самый большой контур
                c = sorted(cnts, key=cv2.contourArea, reverse=True)[0]
                array_x = []
                array_y = []
                for i in range(len(c) - 1):
                    xy = c[i][0]
                    array_x.append(xy[0])
                    array_y.append(xy[1])

                poly = Polygon(list(zip(array_x, array_y)), animated=True,
fill=False, color="blue")
                fig, ax = plt.subplots()
                ax.add_patch(poly)
                p = PolygonInteractor(ax, poly)
                plt.imshow(image)
                plt.show()
                new_points = p.get_poly_points()
                new_points = np.array([[p] for p in new_points])
                global n_points
                n_points=new_points

                self.perspektiva.setEnabled(True)
                self.contur.setEnabled(True)
                self.raschet.setEnabled(True)
                self.conttable.setEnabled(True)
                self.saveprogekt.setEnabled(True)
                self.action.setEnabled(True)
                pass
        else:
            QtWidgets.QMessageBox.information(self, "Ошибка",
                                              "Контур найден некорректно!
Исправте перспективу или загрузите другое изображение")


    # рассчет площади
    def find_cntr(self):
         image = cv2.imread("temp/size_img.jpg")
         edged = cv2.Canny(image, 10, 250)
         kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7))
         closed = cv2.morphologyEx(edged, cv2.MORPH_CLOSE, kernel)
         sharpen = cv2.GaussianBlur(closed, (0, 0), 3)
         sharpen = cv2.addWeighted(closed, 1.5, sharpen, -0.5, 0)
```

```
        cnts, _ = cv2.findContours(sharpen, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        # найдем самый большой контур
        c = sorted(cnts, key=cv2.contourArea, reverse=True)[0]
        sq = cv2.contourArea(c)
        s = sq / 1000000
        text = "Площадь делового остатка равна: "
        text = (text) + str(s) + " м2"
        msgBox = QMessageBox()
        msgBox.setIcon(QMessageBox.Information)
        msgBox.setText(text)
        msgBox.setWindowTitle("Площадь")
        returnValue = msgBox.exec()

   def create_database(self):
       global F
       global n_points
       access = CreateObject('Access.Application')
       DBEngine = access.DBEngine
       db = DBEngine.CreateDatabase((F+'/coordinates.mdb'),
Access.DB_LANG_GENERAL)

       db.BeginTrans()

       db.Execute("CREATE TABLE coordinates (n Integer, x Integer, y
Integer)")

       for i in range(len(n_points)):
           xy = n_points[i][0]
           x = math.ceil(xy[0])
           y = math.ceil(xy[1])
           n = i + 1
           db.Execute("INSERT INTO coordinates VALUES (" + str(n) + "," +
str(x) + "," + str(y) + ")")
       db.CommitTrans()
       db.Close()


   def open_database(self):
       global F
       files = os.listdir(F)
       for the_file in files:
           if the_file == 'coordinates.mdb':
               os.remove(F+'/coordinates.mdb')
       self.create_database()
       os.startfile(F+'/coordinates.mdb')

   def save_project(self):
       global F
       f1=F+'/open_img.jpg'
       f2=F+'/size_img.jpg'
       f3=F+'/transform_img.jpg'
       image1 = cv2.imread("temp/open_img.jpg")
       image2 = cv2.imread("temp/size_img.jpg")
       image3 = cv2.imread("temp/transform_img.jpg")
       cv2.imwrite(f1, image1)
       cv2.imwrite(f2, image2)
       cv2.imwrite(f3, image3)
```

```
        msgBox = QMessageBox()
        msgBox.setIcon(QMessageBox.Information)
        msgBox.setText("Проект сохранен")
        msgBox.setWindowTitle("Проект")
        returnValue = msgBox.exec()

    def show_window_4(self):
        self.w4 = Window4()
        self.w4.show()
#=====================================================================
# Редактировать проект
class progect_R(object):
    def setupUi(self, MainWindow):
        global F1
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(660, 490)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("image/tema.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        MainWindow.setWindowIcon(icon)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.image = QtWidgets.QLabel(self.centralwidget)
        self.image.setGeometry(QtCore.QRect(220, 20, 411, 371))
        self.image.setFrameShape(QtWidgets.QFrame.Box)
        self.image.setText("")
        self.image.setPixmap(QtGui.QPixmap(F1+"/open_img.jpg"))
        self.image.setScaledContents(True)
        self.image.setObjectName("image")
        self.perspektiva = QtWidgets.QPushButton(self.centralwidget)
        self.perspektiva.setGeometry(QtCore.QRect(10, 80, 201, 61))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.perspektiva.setFont(font)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("image/persp.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.perspektiva.setIcon(icon)
        self.perspektiva.setIconSize(QtCore.QSize(20, 20))
        self.perspektiva.setObjectName("perspektiva")
        self.raschet = QtWidgets.QPushButton(self.centralwidget)
        self.raschet.setGeometry(QtCore.QRect(10, 200, 201, 61))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.raschet.setFont(font)
        icon1 = QtGui.QIcon()
        icon1.addPixmap(QtGui.QPixmap("image/area.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.raschet.setIcon(icon1)
        self.raschet.setIconSize(QtCore.QSize(20, 20))
        self.raschet.setObjectName("raschet")
        self.contur = QtWidgets.QPushButton(self.centralwidget)
        self.contur.setGeometry(QtCore.QRect(10, 140, 201, 61))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.contur.setFont(font)
        icon2 = QtGui.QIcon()
```

```
        icon2.addPixmap(QtGui.QPixmap("image/contur.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.contur.setIcon(icon2)
        self.contur.setIconSize(QtCore.QSize(20, 20))
        self.contur.setObjectName("contur")
        self.conttable = QtWidgets.QPushButton(self.centralwidget)
        self.conttable.setGeometry(QtCore.QRect(10, 260, 201, 61))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.conttable.setFont(font)
        icon3 = QtGui.QIcon()
        icon3.addPixmap(QtGui.QPixmap("image/database.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.conttable.setIcon(icon3)
        self.conttable.setIconSize(QtCore.QSize(20, 20))
        self.conttable.setObjectName("conttable")
        self.zakryt = QtWidgets.QPushButton(self.centralwidget)
        self.zakryt.setGeometry(QtCore.QRect(530, 400, 101, 41))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.zakryt.setFont(font)
        icon4 = QtGui.QIcon()
        icon4.addPixmap(QtGui.QPixmap("image/close.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.zakryt.setIcon(icon4)
        self.zakryt.setIconSize(QtCore.QSize(16, 16))
        self.zakryt.setObjectName("zakryt")
        self.saveprogekt = QtWidgets.QPushButton(self.centralwidget)
        self.saveprogekt.setGeometry(QtCore.QRect(400, 400, 121, 41))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.saveprogekt.setFont(font)
        icon5 = QtGui.QIcon()
        icon5.addPixmap(QtGui.QPixmap("image/save.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.saveprogekt.setIcon(icon5)
        self.saveprogekt.setIconSize(QtCore.QSize(20, 20))
        self.saveprogekt.setObjectName("saveprogekt")
        self.saveprogekt.setEnabled(False)
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(10, 20, 201, 61))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.pushButton.setFont(font)
        icon6 = QtGui.QIcon()
        icon6.addPixmap(QtGui.QPixmap("image/loag.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.pushButton.setIcon(icon6)
        self.pushButton.setIconSize(QtCore.QSize(24, 24))
        self.pushButton.setObjectName("pushButton")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 646, 21))
        self.menubar.setObjectName("menubar")
        self.menu = QtWidgets.QMenu(self.menubar)
        self.menu.setObjectName("menu")
        self.menu_2 = QtWidgets.QMenu(self.menubar)
        self.menu_2.setObjectName("menu_2")
```

```
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.action = QtWidgets.QAction(MainWindow)
        self.action.setIcon(icon5)
        self.action.setObjectName("action")
        self.action.setEnabled(False)
        self.action_2 = QtWidgets.QAction(MainWindow)
        self.action_2.setIcon(icon4)
        self.action_2.setObjectName("action_2")
        self.action_3 = QtWidgets.QAction(MainWindow)
        icon7 = QtGui.QIcon()
        icon7.addPixmap(QtGui.QPixmap("image/spravca.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.action_3.setIcon(icon7)
        self.action_3.setObjectName("action_3")
        self.action_4 = QtWidgets.QAction(MainWindow)
        icon8 = QtGui.QIcon()
        icon8.addPixmap(QtGui.QPixmap("image/o_prooect.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.action_4.setIcon(icon8)
        self.action_4.setObjectName("action_4")
        self.menu.addAction(self.action)
        self.menu.addAction(self.action_2)
        self.menu_2.addAction(self.action_3)
        self.menu_2.addAction(self.action_4)
        self.menubar.addAction(self.menu.menuAction())
        self.menubar.addAction(self.menu_2.menuAction())

        self.pushButton.clicked.connect(self.view_img)
        self.perspektiva.clicked.connect(self.persp_view)
        self.zakryt.clicked.connect(self.close)
        self.zakryt.clicked.connect(self.close)
        self.raschet.clicked.connect(self.find_cnt_s)
        self.contur.clicked.connect(self.cnt_view)
        self.conttable.clicked.connect(self.open_database)
        self.saveprogekt.clicked.connect(self.save_project)
        self.action.triggered.connect(self.save_project)
        self.action_2.triggered.connect(self.close)
        self.action_3.triggered.connect(self.show_spravka)
        self.action_4.triggered.connect(self.show_window_4)
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Проект"))
        self.perspektiva.setToolTip(
            _translate("MainWindow", "<html><head/><body><p>Исправить
изображение</p></body></html>"))
        self.perspektiva.setText(_translate("MainWindow", " Перспектива"))
        self.raschet.setToolTip(_translate("MainWindow",
"<html><head/><body><p>Площадь детали</p></body></html>"))
        self.raschet.setWhatsThis(
            _translate("MainWindow", "<html><head/><body><p>Рассчитать
площадь детали</p></body></html>"))
        self.raschet.setText(_translate("MainWindow", " Расчет"))
```

```
        self.contur.setToolTip(
            _translate("MainWindow", "<html><head/><body><p>Вывести контур
детали</p></body></html>"))
        self.contur.setText(_translate("MainWindow", " Контур"))
        self.conttable.setToolTip(
            _translate("MainWindow", "<html><head/><body><p>Открыть базу
данных с координатами</p></body></html>"))
        self.conttable.setText(_translate("MainWindow", "Координаты"))
        self.zakryt.setText(_translate("MainWindow", "Закрыть"))
        self.saveprogekt.setText(_translate("MainWindow", "Сохранить"))
        self.pushButton.setText(_translate("MainWindow", "  Загрузить"))
        self.menu.setTitle(_translate("MainWindow", "Проект"))
        self.menu_2.setTitle(_translate("MainWindow", "Помощь"))
        self.action.setText(_translate("MainWindow", "Сохранить"))
        self.action_2.setText(_translate("MainWindow", "Выход"))
        self.action_3.setText(_translate("MainWindow", "Справка"))
        self.action_4.setText(_translate("MainWindow", "О программе"))

    def show_spravka(self):
        os.startfile(r'User_manual.pdf')

    # загружаем изображение
    def loag_img(self):
        fileName = QtWidgets.QFileDialog.getOpenFileName(self,
                                                         "Выберите файл",
QtCore.QDir.homePath(),
                                                         "Изображение
(*.jpg)")[0]
        return fileName

    def view_img(self):
        global F1
        fileName = self.loag_img()
        if fileName == "":
            QtWidgets.QMessageBox.information(self, "Ошибка",
                                              "Не удалось открыть
изображение")
        else:
            img = cv2.imread(fileName)
            cv2.imwrite('temp/open_img.jpg', img)
            self.image.setPixmap(QtGui.QPixmap("temp/open_img.jpg"))
            self.perspektiva.setEnabled(True)
            self.contur.setEnabled(False)
            self.raschet.setEnabled(False)
            self.conttable.setEnabled(False)
            self.saveprogekt.setEnabled(False)
            self.action.setEnabled(False)

    def persp_view(self):
        # делаем кнопку неактивной
        self.correction_img()
        self.perspektiva.setEnabled(True)
        self.contur.setEnabled(True)
        self.raschet.setEnabled(False)
        self.conttable.setEnabled(False)
        self.saveprogekt.setEnabled(False)
        self.action.setEnabled(False)
        pass
```

```
    # исправляем перспективу
    def correction_img(self):
        global F1
        image = F1+'/open_img.jpg'
        pers = perspectiva()
        pers.obrabotka(image)
        self.image.setPixmap(QtGui.QPixmap(F1+"/transform_img.jpg"))

    def cnt_view(self):
        global F1
        i1 = 'no'
        folder = 'temp/'
        for the_file in os.listdir(folder):
            if the_file == 'transform_img.jpg':
                i1 = 'yes'
        if (i1 == 'yes' ):
            image = cv2.imread('temp/transform_img.jpg')
            image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            image_edge = cv2.Canny(image_gray, 80, 160)

            cnts, hiers = cv2.findContours(image_edge.copy(),
cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)[-2:]
            cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:5]
            screenCnt = []
            for c in cnts:
                peri = cv2.arcLength(c, True)
                approx = cv2.approxPolyDP(c, 0.02 * peri, True)
                if len(approx) == 4:
                    screenCnt = approx
                    break
            # площадь листа A4 62370  A4 - 210x297

            if len(screenCnt) == 4:
                xy1 = screenCnt[0][0]
                xy2 = screenCnt[1][0]
                xy3 = screenCnt[2][0]
                xy4 = screenCnt[3][0]
                x1 = xy1[0]
                y1 = xy1[1]
                x2 = xy2[0]
                y2 = xy2[1]
                x3 = xy3[0]
                y3 = xy3[1]
                x4 = xy4[0]
                y4 = xy4[1]

                d1 = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
                d2 = math.sqrt((x3 - x2) ** 2 + (y3 - y2) ** 2)
                d3 = math.sqrt((x4 - x3) ** 2 + (y4 - y3) ** 2)
                d4 = math.sqrt((x1 - x4) ** 2 + (y1 - y4) ** 2)

                if d1 > d2:
                    width = (d2 + d4) / 2
                    height = (d1 + d3) / 2
                else:
                    width = (d1 + d3) / 2
                    height = (d2 + d4) / 2
```

```
                w = width
                h = height

                image_R = Image.open('temp/transform_img.jpg')
                width_i, height_i = image_R.size
                width_A4 = 210
                height_A4 = 297
                if h > height_A4:
                    h_i = math.ceil(height_i / (h / height_A4))
                else:
                    h_i = math.ceil(height_i * (height_A4 / h))
                if w > width_A4:
                    w_i = math.ceil(width_i / (w / width_A4))
                else:
                    w_i = math.ceil(width_i * (width_A4 / w))
                size_R = (h_i, w_i)
                image_Res = image_R.resize(size_R)
                image_Res.save('temp/size_img.jpg')

                self.image.setPixmap(QtGui.QPixmap("temp/size_img.jpg"))

                image = cv2.imread('temp/size_img.jpg')
                image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
                image_edge = cv2.Canny(image_gray, 80, 160)
                cnts, hiers = cv2.findContours(image_edge.copy(),
cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)[-2:]
                cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:5]
                for c in cnts:
                    peri = cv2.arcLength(c, True)
                    approx = cv2.approxPolyDP(c, 0.02 * peri, True)
                    if len(approx) == 4:
                        screenCnt = approx
                        break
                sq = cv2.contourArea(screenCnt)
                if ((sq >= 65000) or (sq <= 61000)):
                    msgBox = QMessageBox()
                    msgBox.setIcon(QMessageBox.Information)
                    msgBox.setText("Контур найден некорректно! Исправте
перспективу или загрузите другое изображение")
                    msgBox.setWindowTitle("Ошибка")
                    returnValue = msgBox.exec()
                else:
                    image = cv2.imread("temp/size_img.jpg")
                    edged = cv2.Canny(image, 10, 250)
                    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (7,
7))
                    closed = cv2.morphologyEx(edged, cv2.MORPH_CLOSE,
kernel)
                    sharpen = cv2.GaussianBlur(closed, (0, 0), 3)
                    sharpen = cv2.addWeighted(closed, 1.5, sharpen, -0.5,
0)
                    cnts, _ = cv2.findContours(sharpen, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
                    # найдем самый большой контур
                    c = sorted(cnts, key=cv2.contourArea, reverse=True)[0]
                    array_x = []
                    array_y = []
```

```
                    for i in range(len(c) - 1):
                        xy = c[i][0]
                        array_x.append(xy[0])
                        array_y.append(xy[1])

                    poly = Polygon(list(zip(array_x, array_y)),
animated=True, fill=False, color="blue")
                    fig, ax = plt.subplots()
                    ax.add_patch(poly)
                    p = PolygonInteractor(ax, poly)
                    plt.imshow(image)
                    plt.show()
                    new_points = p.get_poly_points()
                    new_points = np.array([[p] for p in new_points])
                    global n_points
                    n_points = new_points

                    self.perspektiva.setEnabled(True)
                    self.contur.setEnabled(True)
                    self.raschet.setEnabled(True)
                    self.conttable.setEnabled(True)
                    self.saveprogekt.setEnabled(True)
                    self.action.setEnabled(True)
                    pass
            else:
                QtWidgets.QMessageBox.information(self, "Ошибка",
                                                  "Контур найден
некорректно! Исправте перспективу или загрузите другое изображение")
        else:
            self.cntr_img()

    # ввывод контура
    def cntr_img(self):
        global F1
        image = cv2.imread(F1+'/size_img.jpg')
        image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        image_edge = cv2.Canny(image_gray, 80, 160)
        cnts, hiers = cv2.findContours(image_edge.copy(), cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)[-2:]
        cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:5]
        for c in cnts:
            peri = cv2.arcLength(c, True)
            approx = cv2.approxPolyDP(c, 0.02 * peri, True)
            if len(approx) == 4:
                screenCnt = approx
                break
        sq = cv2.contourArea(screenCnt)
        if ((sq >= 65000) or (sq <= 61000)):
            msgBox = QMessageBox()
            msgBox.setIcon(QMessageBox.Information)
            msgBox.setText("Контур найден некорректно! Исправте
перспективу или загрузите другое изображение")
            msgBox.setWindowTitle("Ошибка")
            returnValue = msgBox.exec()
        else:
            image = cv2.imread(F1+"/size_img.jpg")
            edged = cv2.Canny(image, 10, 250)
            kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7))
```

```
            closed = cv2.morphologyEx(edged, cv2.MORPH_CLOSE, kernel)
            sharpen = cv2.GaussianBlur(closed, (0, 0), 3)
            sharpen = cv2.addWeighted(closed, 1.5, sharpen, -0.5, 0)
            cnts, _ = cv2.findContours(sharpen, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
            # найдем самый большой контур
            c = sorted(cnts, key=cv2.contourArea, reverse=True)[0]
            array_x = []
            array_y = []
            for i in range(len(c) - 1):
                xy = c[i][0]
                array_x.append(xy[0])
                array_y.append(xy[1])

            poly = Polygon(list(zip(array_x, array_y)), animated=True,
fill=False, color="blue")
            fig, ax = plt.subplots()
            ax.add_patch(poly)
            p = PolygonInteractor(ax, poly)
            plt.imshow(image)
            plt.show()
            new_points = p.get_poly_points()
            new_points = np.array([[p] for p in new_points])
            global n_points
            n_points = new_points

            self.perspektiva.setEnabled(True)
            self.contur.setEnabled(True)
            self.raschet.setEnabled(True)
            self.conttable.setEnabled(True)
            self.saveprogekt.setEnabled(True)
            self.action.setEnabled(True)
            pass

    def find_cnt_s(self):
        i1 = 'no'
        folder = 'temp/'
        for the_file in os.listdir(folder):
            if the_file == 'size_img.jpg':
                i1 = 'yes'
        if (i1 == 'yes'):
            image = cv2.imread("temp/size_img.jpg")
            edged = cv2.Canny(image, 10, 250)
            kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7))
            closed = cv2.morphologyEx(edged, cv2.MORPH_CLOSE, kernel)
            sharpen = cv2.GaussianBlur(closed, (0, 0), 3)
            sharpen = cv2.addWeighted(closed, 1.5, sharpen, -0.5, 0)
            cnts, _ = cv2.findContours(sharpen, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
            # найдем самый большой контур
            c = sorted(cnts, key=cv2.contourArea, reverse=True)[0]
            sq = cv2.contourArea(c)
            s = sq / 1000000
            text = "Площадь делового остатка равна: "
            text = (text) + str(s) + " м2"
            msgBox = QMessageBox()
            msgBox.setIcon(QMessageBox.Information)
            msgBox.setText(text)
```

```
            msgBox.setWindowTitle("Площадь")
            returnValue = msgBox.exec()
        else:
            self.find_cntr()

    # рассчет площади
    def find_cntr(self):
        global F1
        image = cv2.imread(F1+"/size_img.jpg")
        edged = cv2.Canny(image, 10, 250)
        kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7))
        closed = cv2.morphologyEx(edged, cv2.MORPH_CLOSE, kernel)
        sharpen = cv2.GaussianBlur(closed, (0, 0), 3)
        sharpen = cv2.addWeighted(closed, 1.5, sharpen, -0.5, 0)
        cnts, _ = cv2.findContours(sharpen, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        # найдем самый большой контур
        c = sorted(cnts, key=cv2.contourArea, reverse=True)[0]
        sq = cv2.contourArea(c)
        s = sq / 1000000
        text = "Площадь делового остатка равна: "
        text = (text) + str(s) + " м2"
        msgBox = QMessageBox()
        msgBox.setIcon(QMessageBox.Information)
        msgBox.setText(text)
        msgBox.setWindowTitle("Площадь")
        returnValue = msgBox.exec()

    def create_database(self):
        global F1
        global n_points
        access = CreateObject('Access.Application')
        DBEngine = access.DBEngine
        db = DBEngine.CreateDatabase((F1 + '/coordinates.mdb'),
Access.DB_LANG_GENERAL)

        db.BeginTrans()

        db.Execute("CREATE TABLE coordinates (n Integer, x Integer, y
Integer)")

        for i in range(len(n_points)):
            xy = n_points[i][0]
            x = math.ceil(xy[0])
            y = math.ceil(xy[1])
            n = i + 1
            db.Execute("INSERT INTO coordinates VALUES (" + str(n) + "," +
str(x) + "," + str(y) + ")")
        db.CommitTrans()
        db.Close()

    def open_database(self):
        global F1
        global n_points
        if (len(n_points)==0):
            os.startfile(F1 + '/coordinates.mdb')
        else:
            files = os.listdir(F1)
```

```
            for the_file in files:
                if the_file == 'coordinates.mdb':
                    os.remove(F1 + '/coordinates.mdb')
            self.create_database()
            os.startfile(F1 + '/coordinates.mdb')

    def save_project(self):
        global F1
        i1='no'
        i2='no'
        i3='no'
        folder = ('temp/')
        for the_file in os.listdir(folder):
            if the_file == 'open_img.jpg':
                i1 = 'yes'
            if the_file == 'size_img.jpg':
                i2 = 'yes'
            if the_file == 'transform_img.jpg':
                i3 = 'yes'
        if (i1 == 'yes' ):
            f1 = F1 + '/open_img.jpg'
            image1 = cv2.imread('temp/open_img.jpg')
            cv2.imwrite(f1, image1)
        if (i2 == 'yes' ):
            f2 = F1 + '/size_img.jpg'
            image2 = cv2.imread('temp/size_img.jpg')
            cv2.imwrite(f2, image2)
        if (i3 == 'yes' ):
            f3 = F1 + '/transform_img.jpg'
            image3 = cv2.imread('temp/transform_img.jpg')
            cv2.imwrite(f3, image3)

        msgBox = QMessageBox()
        msgBox.setIcon(QMessageBox.Information)
        msgBox.setText("Проект сохранен")
        msgBox.setWindowTitle("Проект")
        returnValue = msgBox.exec()

    def show_window_4(self):
        self.w4 = Window4()
        self.w4.show()
#=====================================================================
# о программе
class o_prog(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(562, 194)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("image/tema.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        MainWindow.setWindowIcon(icon)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(90, 130, 381, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.label.setFont(font)
```

```
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(20, 50, 611, 51))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.label_2.setFont(font)
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(230, 20, 91, 21))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.label_3.setFont(font)
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(self.centralwidget)
        self.label_4.setGeometry(QtCore.QRect(150, 10, 51, 51))
        self.label_4.setText("")
        self.label_4.setPixmap(QtGui.QPixmap("image/spravka.png"))
        self.label_4.setScaledContents(True)
        self.label_4.setObjectName("label_4")
        self.label_5 = QtWidgets.QLabel(self.centralwidget)
        self.label_5.setGeometry(QtCore.QRect(20, 90, 251, 20))
        font = QtGui.QFont()
        font.setPointSize(10)
        self.label_5.setFont(font)
        self.label_5.setObjectName("label_5")
        self.label_6 = QtWidgets.QLabel(self.centralwidget)
        self.label_6.setGeometry(QtCore.QRect(170, 0, 51, 61))
        self.label_6.setText("")
        self.label_6.setPixmap(QtGui.QPixmap("image/tema.png"))
        self.label_6.setScaledContents(True)
        self.label_6.setObjectName("label_6")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 562, 21))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "О программе"))
        self.label.setText(_translate("MainWindow", "(c) Зайникова В. Р.,
Зыкин С. А., Петренко А. А.   2019-2020 г. "))
        self.label_2.setText(
            _translate("MainWindow", "Программное обеспечение получения
размеров делового остатка нестандартной формы"))
        self.label_3.setText(_translate("MainWindow", "Imgcoord"))
        self.label_5.setText(_translate("MainWindow", "по цифровой
фотографии."))
#=====================================================================
# для окна о программе 4
class Window4(QMainWindow, o_prog):
   def __init__(self, parent=None):
```

```
        super(Window4, self).__init__(parent)

        self.setupUi(self)
        self.setWindowTitle('О программе')
#=========================================================================
# для окна проект 2
class Window2(QMainWindow, progect):
   def __init__(self, parent=None):
        super(Window2, self).__init__(parent)
        self.setupUi(self)
        self.setWindowTitle('Проект')
#=========================================================================#
для окна редактировать проект 3

class Window3(QMainWindow, progect_R):
    def __init__(self, parent=None):
        super(Window3, self).__init__(parent)
        self.setupUi(self)
        self.setWindowTitle('Редактировать проект')

#=========================================================================
# Открытие окон главное окно
class MyWidget(QMainWindow, Ui_MainWindow):
    def __init__(self):
        super(MyWidget, self).__init__()
        self.setupUi(self)
        self.setWindowTitle('Главное окно')
        # нажатие кнопок на главном окне
        self.pushButton_2.clicked.connect(self.show_window_2)
        self.pushButton_4.clicked.connect(self.show_window_4)
        self.pushButton.clicked.connect(self.show_window_3)
        self.action.triggered.connect(self.show_window_2)
        self.action_4.triggered.connect(self.show_window_4)
        self.action_5.triggered.connect(self.show_window_3)

    # открытие окна Проект
    def show_window_2(self):
        global F
        F = QtWidgets.QFileDialog.getExistingDirectory(self, "Создать
проект в папке", ".")
        if F == "":
            QtWidgets.QMessageBox.information(self, "Ошибка",
                                              "Не выбрана папка для
создания проекта")
        else:
            folder = F
            for the_file in os.listdir(folder):
                if the_file == 'open_img.jpg':
                    os.remove(F + '/open_img.jpg')
                if the_file == 'size_img.jpg':
                    os.remove(F + '/size_img.jpg')
                if the_file == 'transform_img.jpg':
                    os.remove(F + '/transform_img.jpg')
            self.w2 = Window2()
            self.w2.show()

        folder = 'temp/'
        for the_file in os.listdir(folder):
```

```
            file_path = os.path.join(folder, the_file)
            if os.path.isfile(file_path):
                os.unlink(file_path)

    # открытие окна о программе
    def show_window_4(self):
        self.w4 = Window4()
        self.w4.show()

    def show_window_3(self):
        global F1
        image1 = "no"
        image2 = "no"
        image3 = "no"
        F1 = QtWidgets.QFileDialog.getExistingDirectory(self, "Открыть
проект", ".")
        if F1 == "":
            QtWidgets.QMessageBox.information(self, "Ошибка",
                                              "Не выбрана папка для
редактирования проекта")
        else:
            files = os.listdir(F1)
            for the_file in files:
                if the_file == 'open_img.jpg':
                    image1 = "yes"
                if the_file == 'size_img.jpg':
                    image2 = "yes"
                if the_file == 'transform_img.jpg':
                    image3 = "yes"
            if (image1 == 'yes' and image2 == 'yes' and image3 == 'yes'):
                self.w3 = Window3()
                self.w3.show()
            else:
                QtWidgets.QMessageBox.information(self, "Ошибка",
                                                  "В выбраннойпапке не
содержится проект")
        folder = 'temp/'
        for the_file in os.listdir(folder):
            file_path = os.path.join(folder, the_file)
            if os.path.isfile(file_path):
                os.unlink(file_path)

#=====================================================================
if __name__ == "__main__":
    app = QApplication(sys.argv)
    ex = MyWidget()
    ex.show()
    sys.exit(app.exec_())
```