ВВЕДЕНИЕ

Человек восемьдесят процентов информации об окружающем мире воспринимает при помощи зрения. Незначительную часть из всей воспринимаемой информации человек способен осознать. Ещё меньшую часть он способен формализовать, что бы её смог понять и проанализировать компьютер. Текущие системы принятия решений, экспертные системы, при принятии решения руководствуются непосредственно той незначительной частью всей информации, которую смог формализовать для них человек. Громадная часть информации так и остается лежать в виде картинок, различных видео данных. Использование данного рода информации позволит перейти на более качественный уровень работы систем принятия решений и экспертных систем. Возможность рассматривать изображение не как множество точек, а как совокупность объектов позволит дать компьютеру зрение.

Компьютерное зрение – это одна из самых востребованных областей развития цифровых компьютерных технологий. Она требуется на производстве, при управлении роботами, при автоматизации процессов, в медицинских и военных приложениях, при наблюдении со спутников.

В основе научной концепции точного земледелия лежат представления о существовании неоднородностей в пределах одного поля. Для оценки и детектирования этих неоднородностей используются новейшие технологии, такие как системы глобального позиционирования (GPS, ГЛОНАСС), специальные датчики, аэрофотоснимки и снимки со спутников, а также специальные программы для агроменеджмента на базе геоинформационных систем (ГИС). Собранные данные используются для планирования высева, расчёта норм внесения удобрений и средств защиты растений (СЗР), более точного предсказания урожайности и финансового планирования.

Целью дипломного проекта является разработка системы классификации поражений сельскохозяйственной продукции. Для достижения указанной цели в дипломном проекте ставятся следующие задачи:

- проанализировать технологии точного земледелия, существующие коммерческие реализации;
- спроектировать систему классификации поражений сельскохозяйственной продукции;
- провести тестирование разрабатываемой системы, дать анализ эффективности работы системы классификации изображений;
- оформить пояснительную записку.

Реализованная в данном дипломном проекте система может являться часть многофункционального комплекса точного земледелия. Система занимает определенное место и выполняет поставленную задачу. Подобным комплексом является программный пакет Precision Farming Box (PF-Box), облегчающий процесс выполнения работ по точному земледелию. В его состав входят различные модули, в включающие в себя систему контроля состояния растений, по назначению схожая с программным средством дипломного проекта. Она устанавливается на терминале агронома. Это позволяет использовать программу управления сенсором параллельно к другим процессам, например передача данных в центральный офис или сохранение и работа с данными. Агроном может использовать несколько опций программы.

Программное средство так же может использоваться в комплексе с другими программами, работая параллельно с ними, или работать самостоятельно. Программа получает исходные данные из файла изображения, который используют другие программы, сохраняя его на жестком диске компьютера.

1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Актуальность точного земледелия

В последнее десятилетие внимание аграрной науки всего мира привлечено к проблеме разработки методологии, так называемого, «точного» земледелия (precision farming). В США, например, около 4% фермеров используют ее при производстве сельскохозяйственной продукции, а в хозяйствах зернового направления – более 15%. Внедрение передовых технологий в сельскохозяйственном производстве - одно из условий получения субсидий в Европейском Союзе.

По единодушному мнению ученых, переход с директивного на «точное» земледелие позволит оптимизировать производство, обеспечить максимальную прибыль, рационально использовать природные ресурсы, создать условия для надлежащей охраны окружающей среды, поставить систему земледелия на новый, качественный уровень, не изменив его сути.

Основу «точного» земледелия составляют информационные (Information Technology) и геоинформационные технологии (Geographical Information Systems), которые на современном уровне своего развития открывают путь к существенному совершенствованию методов принятия решений в мелиорации и агрономии для формирования высоких урожаев путем интегрированного управления гео- и биофизическими процессами в соответствии с технологическим регламентом и в диалектическом единстве с условиями внешней среды – природными (климат, почва) и хозяйственными (мелиорация, агротехника). Эта информация подчёркнута из книги В.П. Якушева [1], в которой изложена необходимая информация по точному земледелию.

Таким образом, точное земледелие – это неотъемлемая часть современного ведения сельского хозяйства, так как использует в своей работе современные технологии и позволяет оперативно реагировать на состояние сельскохозяйственной продукции.

1.2 Технология точного земледелия

Варианты технологий точного земледелия описаны в книге И.М. Михайленко [2].

Точное земледелие в настоящее время получает все большее распространение во многих странах. Технология точного земледелия рассматривает каждое сельскохозяйственное поле как неоднородное по

рельефу, почвенному покрову, агрохимическому содержанию и подразумевает применение на каждом участке поля разных агротехнологий.

В зависимости от биологической потребности растений, полученной на основании объективных данных, вносится дифференцированная, строго нормированная доза удобрения и только на тех участках поля, где это необходимо. Таким образом, достигается оптимизация питания и обработки всех растений. Это приводит к экономии удобрений и не создает реальной опасности загрязнения окружающей среды. Изображение технологи точного земледелия представлена на схеме ГУИР.400201.005.

Технологии точного земледелия направлены на повышение производительности, уменьшение себестоимости продукции и сохранение окружающей среды. Этого добиваются с помощью целого ряда современных информационных технологий. Среди них главнейшими являются: технология оценки урожайности (Crop Monitor), позволяющая подсчитывать нажин с каждого участка поля; технология глобального позиционирования (Global Positioning System — GPS), когда определяются точные географические координаты каждого участка поля и местонахождение сельскохозяйственной техники и механизмов; технология так называемого сменного нормирования (Variable Rate Technology — VRT), когда в зависимости от ситуации на каждом отдельном участке поля выполняют (или опускают) необходимую технологическую операцию.

В общем виде технология, представленная на рисунке 1.1, выглядит следующим образом: спутник с высоты сантиметр за сантиметром сканирует сельскохозяйственные площади. В зависимости от палитры цветов узнают о содержании азота, фосфора, калия, о кислотности почвы. Фермер получает химическую «картину» собственного поля. Потом закладывает картограмму в компьютер, руководящий агрегатами во время сева. При прохождении по полю специальная программа считывает космическое изображение, идентифицирует с реальной площадью и там, где встречаются истощенные участки, через сошники автоматически обогащает их соответствующими минеральными удобрениями. Космическое обследование сельхозугодий проводят раз в четыре года.

Эта технология универсальна для различных способов контроля за состоянием на поле. То есть, схожим способом проходит загрузка, проверка, контроль состояния растений по их изображением. Дальнейшие действия принимает агроном, если программа работает отдельно, или другая программа-анализатор, которая делает выводы и отдаёт команды на устройства

сельхозтехники, анализируя сохранённые изображения, если классификатор поражений дипломного проекта работает в комплексе.

Рисунок 1.1 – Упрощенная схема точного земледелия

Ядром комплекса управления технологии точного земледелия является система поддержки принятия решений (СППР). Она формирует так называемые карты обработки (treat-ment maps), которые определяют, как следует обрабатывать каждую единицу управления на сельскохозяйственном поле.

Электронная карта обработки (chip card) загружается в робототехническое устройство, находящиеся на сельскохозяйственном агрегате.

Технология точного земледелия включает следующие этапы работы:

- Создание электронных карт полей.
- Создание базы данных по полям (площадь, урожайность, агрохимические и агрофизические свойства фактические и нормативные, уровень развития растений и т.д.).
- Проведение анализа в программном обеспечении и выдача наглядных форм для выработки решений.
- Выдача команд по принимаемым решениям на чип-картах, которые загружаются в робототехнические устройства на сельскохозяйственные агрегаты для дифференцированного проведения обработки растений.

Для работы по технологии точного земледелия необходимо:

- разбить поле на единицы управления - квадраты, которые имеют одинаковые площади, удобные для обработки агрегатами, имеют собственные номера и считаются однородными элементарными участками (одинаковыми по

почвенным характеристикам, содержанию питательных веществ, каменистостью и другим параметрам) с пространственной привязкой к местности;

- отобрать почвенные пробы с пространственной привязкой к местности;
- определить содержание питательных веществ по каждой единице управления
- и построить карту распределения агрохимических показателей;
- обработать, проанализировать с помощью программного средства и составить технологическую карту дифференцированного внесения удобрений.

Точное земледелие предполагает подробное первичное агрохимическое обследование полей. В дальнейшем анализируются карты урожайности, что позволяет значительно уменьшить количество проб. Последовательность и этапы классификации поражений сельскохозяйственной продукции показаны на схеме ГУИР.400201.006.

1.3 Методы обработки изображений. Сегментация

В качестве источника информации о методах обработки изображений были использованы книги Г. Вудса [3] и Б. Яне [4]. Они дают необходимое представление о способах обработки изображений и предлагают необходимые определения понятий по теме.

Сегментация — это процесс разделения цифрового изображения на несколько сегментов (множество пикселей, также называемых суперпикселями). Цель сегментации заключается в упрощении и/или изменении представления изображения, чтобы его было проще и легче анализировать. Сегментация изображений обычно используется для того, чтобы выделить объекты и границы (линии, кривые, и т. д.) на изображениях.

Результатом сегментации изображения является множество сегментов, которые вместе покрывают всё изображение, или множество контуров, выделенных из изображения. Все пиксели в сегменте похожи по некоторой характеристике или вычисленному свойству, например по цвету, яркости или текстуре. Соседние сегменты значительно отличаются по этой характеристике.

Существует несколько способов сегментации:

- Методы, основанные на кластеризации. k-средних — это итеративный метод, который используется, чтобы разделить изображение на K кластеров. Реализация базового алгоритма:

- выбрать K центров кластеров, случайно или на основании некоторой эвристики;

- поместить каждый пиксель изображения в кластер, центр которого ближе всего к этому пикселю;

- заново вычислить центры кластеров, усредняя все пиксели в кластере;

- повторять шаги 2 и 3 до сходимости.

- Методы с использованием гистограмм. Методы с использованием гистограммы очень эффективны, когда сравниваются с другими методами сегментации изображений, потому что они требуют только один проход по пикселям. В этом методе гистограмма вычисляется по всем пикселям изображения и её минимумы и максимумы используются, чтобы найти кластеры на изображении. Цвет или яркость могут быть использованы при сравнении. Улучшение этого метода — рекурсивно применять его к кластерам на изображении для того, чтобы поделить их на более мелкие кластеры.

- Выделение краёв. Выделение краёв — это хорошо изученная область в обработке изображений. Границы и края областей сильно связаны, так как часто существует сильный перепад яркости на границах областей. Поэтому методы выделения краёв используются как основа для другого метода сегментации. Обнаруженные края часто бывают разорванными. Но чтобы выделить объект на изображении, нужны замкнутые границы области.

1.4 Основные признаки заболевания растений

Из работы Е.Н. Санкиной [5] подчёркнуты сведения о заболеваниях сельскохозяйственных растений, характерные признаки, способы лечения.

Фитопатология — наука о болезнях растений, вызванных патогенами (инфекционные болезни) и экологическими факторами (физиологические факторы). Включает разработку средств борьбы с заболеваниями, профилактику поражения растений.

Основная цель выделения признаков – переход из пространства образов в пространство признаков, имеющее значительно меньшую размерность. Признаки должны обеспечивать компактность и желательно линейную разделимость классов в пространстве признаков. То есть, каждый образ должен быть близок к образам своего класса, и удален от других классов.

Качество информативных признаков в конечном итоге определяется в результате классификации на обучающей выборке. Если сеть разбивает обучающую выборку на кластеры, число которых равно числу классов, и в

каждый кластер отображаются только образы одного класса, то можно говорить о хорошем разделении классов и об удачном выборе информативных признаков.

Можно выделить три группы поражения растений:

- Растения, зараженные болезнью из группы alternaria (грибные болезни). У пораженных растений на листьях и стеблях, как на рисунке 1.2, появляются тёмно-бурые пятна, увеличивающиеся в размерах. Листья вянут, желтеют и чернеют, затем засыхают, во влажную погоду загнивают, опадают, а стебли надламываются.

Рисунок 1.2 – Поражённые грибковой болезнью листья.

- Растения, зараженные бактериальной болезнью erwinia. Здесь, как на рисунке 1.3, у пораженных растений листья желтеют и свертываются. Нижняя часть стебля и корни загнивают и становятся черными.

Рисунок 1.3 – Растение, поражённое бактериальной болезнью

- Здоровые растения, изображены на рисунке 1.4. Они характеризуются однородным зелёным цветом без участков поражений и высохших листьев.

Рисунок 1.4 – Пример здорового растения

1.5 Программные средства для написания работы

В качестве основного средства для работы с изображениями была выбрана библиотека OpenCV. Это библиотека компьютерного зрения с открытым исходным кодом — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++.

OpenCV написана на языке высокого уровня и содержит алгоритмы для интерпретации изображений, калибровки камеры по эталону, устранение оптических искажений, определение сходства, анализ перемещения объекта, определение формы объекта и слежение за объектом, 3D-реконструкция, сегментация объекта, распознавание жестов и т.д. OpenCV был разработан для обеспечения эффективности вычислений с сильным акцентом на приложения реального времени. Эта библиотека очень популярна за счёт своей открытости и возможности бесплатно использовать как в учебных, так и коммерческих целях.

С момента выпуска альфа-версии в январе 1999 года, OpenCV был использован во многих приложениях, продуктах и научно-исследовательских работах. Эти приложения включают приложения по работе со сшитыми воедино образами со спутников и веб-карт, выравниванием сканированных изображений, уменьшением шума медицинских изображений, анализом объектов, безопасностью и системами обнаружения вторжений, автоматическим мониторингом и системами безопасности, системами производственного контроля, калибровкой камер, военным применением беспилотных, воздушных, наземных и подводных аппаратах. Библиотека

OpenCV имеет огромный функционал для разработки приложений компьютерного зрения. Компьютерное зрение нашло свое применение в разных областях, начиная от систем безопасности и заканчивая военными роботами.

Общая структура OpenCV представлена на рисунке 1.5

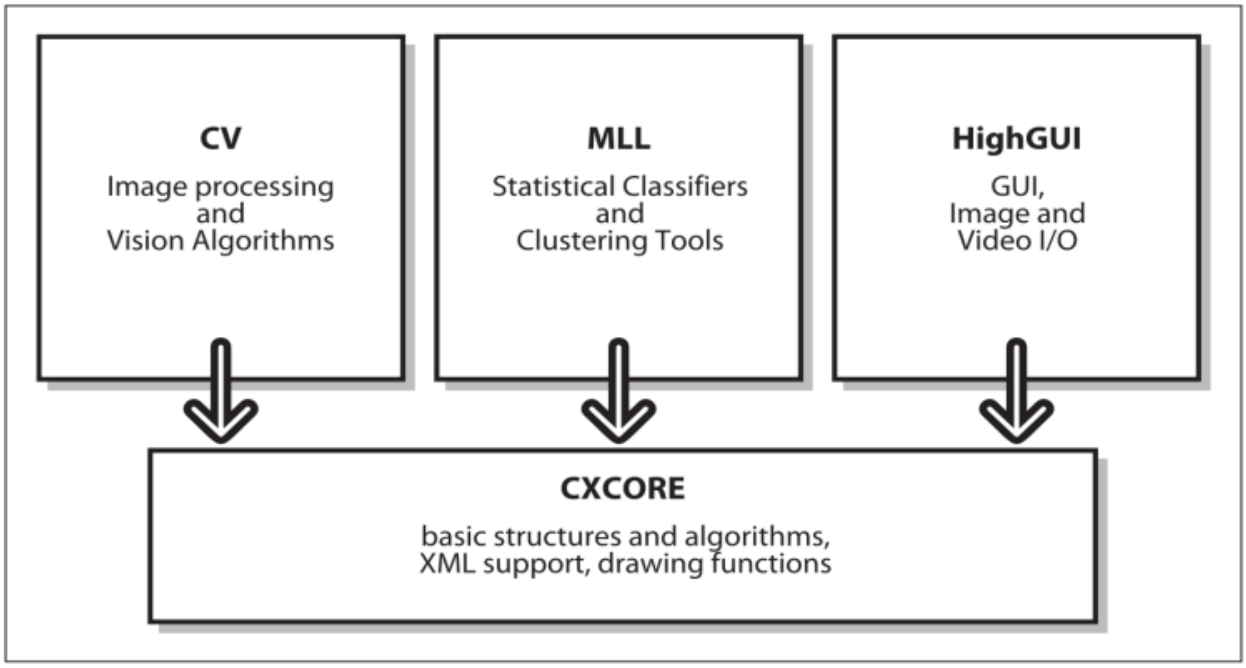

Рисунок 1.5 – Общая структура OpenCV

Вся библиотека разделена по функциональному использованию на подразделы:

- *opencv_core* — ядро: базовые с труктуры, вычисления (математические функции, генерация псевдослучайных чисел, DFT, DCT, ввод/вывод в XML и т.п.);
- *opencv_imgproc* — обработка изображений (фильтры, преобразования и т. д.);
- *opencv_highgui* — простой UI, загрузка/сохранение изображений и видео;
- *opencv_ml* — методы и модели машинного обучения (SVM, деревья принятия решений и т. д.);
- *opencv_features2d* — различные дескрипторы (SURF);
- *opencv_video* — анализ движения и отслеживание объектов (оптический поток, шаблоны движения, устранение фона);
- *opencv_objdetect* — детектирование объектов на изображении (вейвлеты Хаара, HOG и т. д.);
- *opencv_calib3d* — калибровка камеры, поиск стереосоответстсвия и

элементы обработки трехмерных данных;

- *opencv_flann* — библиотека быстрого поиска ближайших соседей;
- *opencv_contrib* — сопутствующий код, еще не готовый для применения;
- *opencv_legacy* — устаревший код, сохраненный ради обратной совместимости;
- *opencv_gpu* — ускорение некоторых функций OpenCV за счет CUDA (NVidia).

В дипломной работе были использованы библиотеки *opencv_imgproc* и *opencv_highgui*. Они содержат функции, необходимые для реализации возможностей программного средства классификации поражений сельскохозяёственной продукции.

В качестве источника информации о работе с OpenCV были выбраны книга Г. Брадски [6] и электронный ресурс [7].

В качестве среды разработки используется Microsoft Visual Studio Express 2010. Microsoft Visual Studio — линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах. для всех платформ, поддерживаемых Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework и Microsoft Silverlight.

Используемый язык для написания проекта – C++. C++ — компилируемый статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование, обобщенное программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

Сведения о особенностях работы в среде Visual Studio 2010 и вспомогательные данные о языке C++ взяты из книги Б. Пахомова [8] и Х. Дейтела [9] соответственно. В этой литературе имеются материалы как по общим понятиям языка программирования, так и более сложные способы решения различных задач, что пригодилось в реализации программной системы дипломного проекта.

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

2.1 Общие требования к системе

Перед разработкой программной системы классификации изображений для мониторинга состояния сельскохозяйственной продукции необходимо определить требования к разрабатываемой системе. Система должна обеспечивать:

- взаимодействие с пользователем;
- уметь работать с изображением, проводить над ним необходимые преобразования;
- осуществлять классификацию полученных результатов.

Поэтому программное средство должно состоять из следующих основных блоков, как на рисунке 2.1: подсистема пользовательского интерфейса, подсистема работы с изображением, подсистема классификации. Структурная схема представлена на схеме ГУИР.400201.001 ПД. На ней отражены подсистемы для каждой решаемой задачи и необходимые модули, входящие в состав отдельной подсистемы.

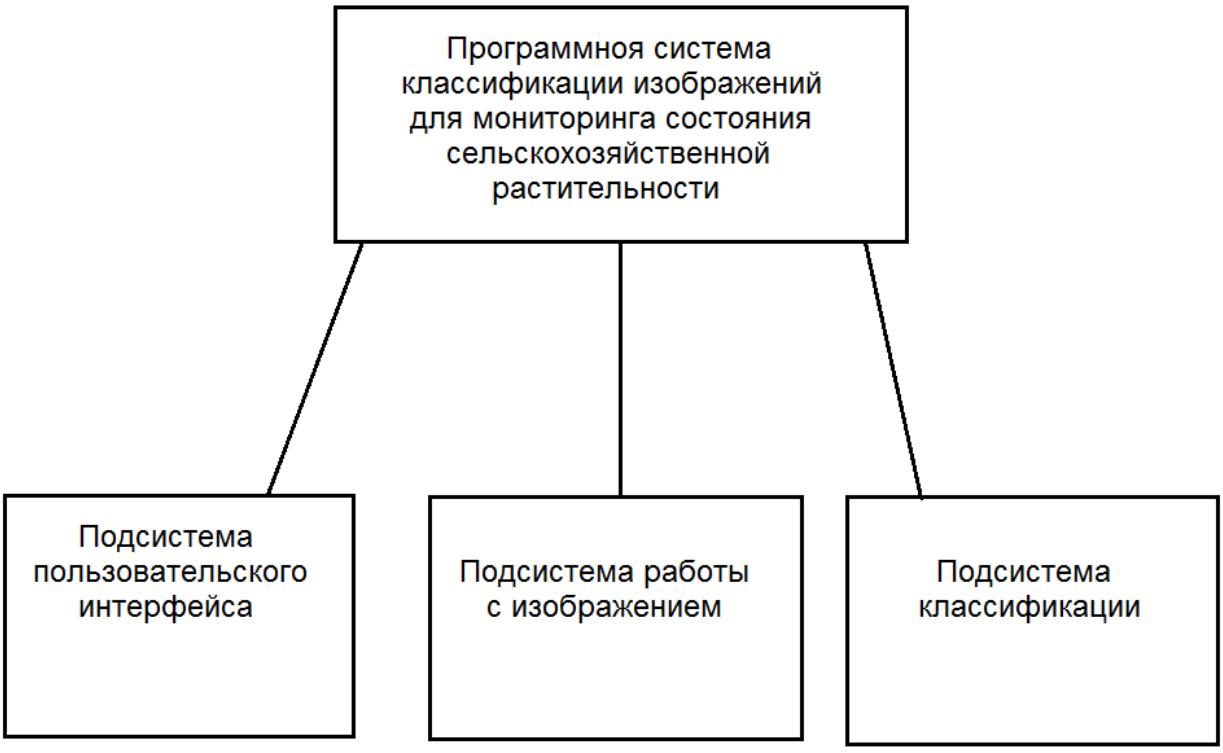

Рисунок 2.1 – Общая структура программного средства

2.2 Подсистема пользовательского интерфейса

Пользовательский интерфейс необходим программе, чтобы взаимодействовать с пользователем. Основная часть программы – окно, на котором расположены необходимые элементы управления и работы с входными и выходными данными. Поэтому подсистема пользовательского интерфейса содержит два модуля: модуль пользовательского интерфейса и модуль обработки входной и выходной информации. Первый выполняет функции построения окна взаимодействия с пользователем, необходимых элементов управления. Второй отвечает за вывод на экран входной информации – изображения исследуемой растительности, и выходной – изображения поражённого участка и его классификация.

Таким образом, подсистема пользовательского интерфейса отвечает за построение окна взаимодействия с пользователем, отображение необходимых элементов управления и вывод на экран исходного изображения и полученного. Выполнение этих требований позволит построить интерфейс взаимодействия с пользователем и не способствует перегрузке рабочего окна ненужными данными и элементами управления. То есть, подсистема позволяет получить интерфейс с необходимым набором элементов управления и отображения. На рисунке 3.1 показана структура подсистемы.

Рисунок 2.2 – Структура подсистемы пользовательского интерфейса.

2.3 Подсистема работы с изображением

Основная работа программного средства дипломного проекта – обработка и анализ входных изображений исходя из решаемой задачи. Первоочередные задачи программы, это выделение объекта и определения цвета. Поэтому подсистему должна содержать два модуля: определения цвета и выделение объекта, что представлено на рисунке 2.3.

Рисунок 2.3 – Структура подсистемы работы с изображением

Модуль выделения объекта должен определять границу объекта по заданным параметрам и отделить его от остального изображения для дальнейшего анализа. При этом результат работы модуля сохраняется в промежуточном изображении для удобства дальнейшей работы самой программы. Эти результаты не доступны для пользователя, поэтому не имеют никакой связи с модулем пользовательского интерфейса.

Модуль определения цвета должен определять необходимый цвет из палитры остальных цветов изображения. Для этого в нём реализованы фильтры работы с изображением. В процессе работы данного модуля также имеют место промежуточные результаты, необходимые для дальнейшей работы программы. Поэтому модуль включает в себя функцию сохранения изображения и его загрузки.

Таким образом, подсистема работы с изображением состоит из двух ключевых модулей, которые, в свою очередь, обладают необходимым набором функций для своей работы. Всё это реализовано в конечном программном средстве, кА необходимая составляющая часть его работы.

2.4 Подсистема классификации

Структура подсистемы классификации представлена на рисунке 2.4.

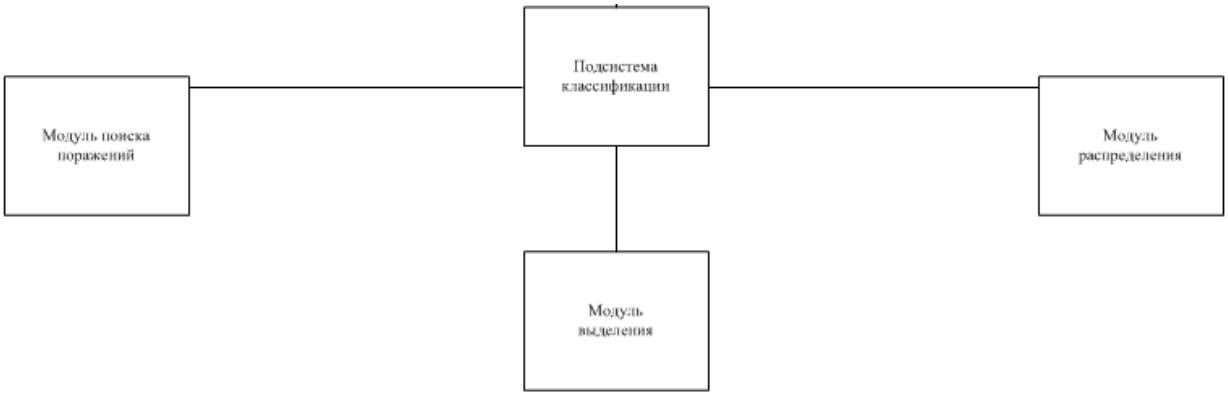

Рисунок 2.4 – Структура подсистемы классификации

Подсистема классификации отвечает за распределение полученных данных от предыдущей подсистемы предоставления результатов работы системы. Состоит из трёх модулей: поиск поражения, выделение и распределение.

Первый модуль отвечает за поиск на изображении необходимого цвета. Поиск осуществляется по заданным параметрам с учётом характеристик поражений растительности.

Модуль выделения служит для отделения на изображении поражённых участков для дальнейшей их обработки. Здесь определяются границы поражения, его положение и выделение на фоне остального содержимого.

Третий модуль – модуль распределения, является классификатором поражений. Он получает данные от двух других блоков, обрабатывает её и выдаёт результат об исходном изображении относительно поставленной задачи. Результат отображается на экране монитора с определением типа поражения. Данный модуль является конечным в работе программы и в ходе обработки изображения.

Таким образом, программное средство классификации поражений состоит из трёх основных подсистем, которые обеспечивают работу программы по поставленной задаче, не перегружают систему. Взаимодействие происходит параллельно решаемым задачам между подсистемами и внутри каждой подсистемы – между блоками.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

3.1 Реализация основных модулей системы

На данном этапе были реализованы основные алгоритмы программного средства классификации поражений для мониторинга состояния сельскохозяйственной продукции. Функциональное проектирование нацелено, прежде всего, на создание эффективно работающего объекта. Выполнение требуемой функции — главная цель и основа разработки объекта. Здесь будут подробнее рассмотрены основные модули программного средства.

3.2 Диаграмма объектов

Диаграмма объектов - диаграмма, на которой показаны взаимодействия объектов, упорядоченные по времени их проявления.

Диаграмма объектов показывает основные классы, их содержание, выполняемые операции и связи между ними. Диаграмма объектов представлена на схеме ГУИР.400201.004 ПД.

Основной элемент – рабочее окно, на котором отображены все необходимые элементы управления: рабочее пространство, панель инструментов, меню. Для построения рабочего окна используется класс CWinApp.

Класс операции с файлами необходим для обеспечения основных действий с файлами: открыть файл и сохранить.

Класс отображения изменений выполняет функцию показа изменённого изображения. Этот класс взаимодействует с классом рабочего окна, так как в последнем отображает вся конечная информация.

Класс обработки изображения включает в себя функции преобразования изображения, определения контуров и сегментацию. Для этого ему необходимы классы изображения и окна изображения, которые взаимодействуют с основным классом – рабочее окно.

Класс изображения предназначен для предварительной работы с изображением: получение размера и начальных точек. Для этого ему необходимо взаимодействие с классом операций с файлами, где происходит загрузка изображения.

Классификация необходима для распределения результатов и предоставления конечного ответ по поставленной задаче – классификации поражений растительности.

Все классы имеют связь с основным – рабочее окно, так как результаты работы должны отображаться на экране.

3.3 Реализация модуля пользовательского интерфейса

Графический пользовательский интерфейс реализован в модуле пользовательского интерфейса. Данный модуль предоставляет пользователю возможность настройки основных алгоритмов, которые используются для слежения за объектом. Диаграмма классов, отражающая структуру реализованного модуля, представлена на рисунке 3.1

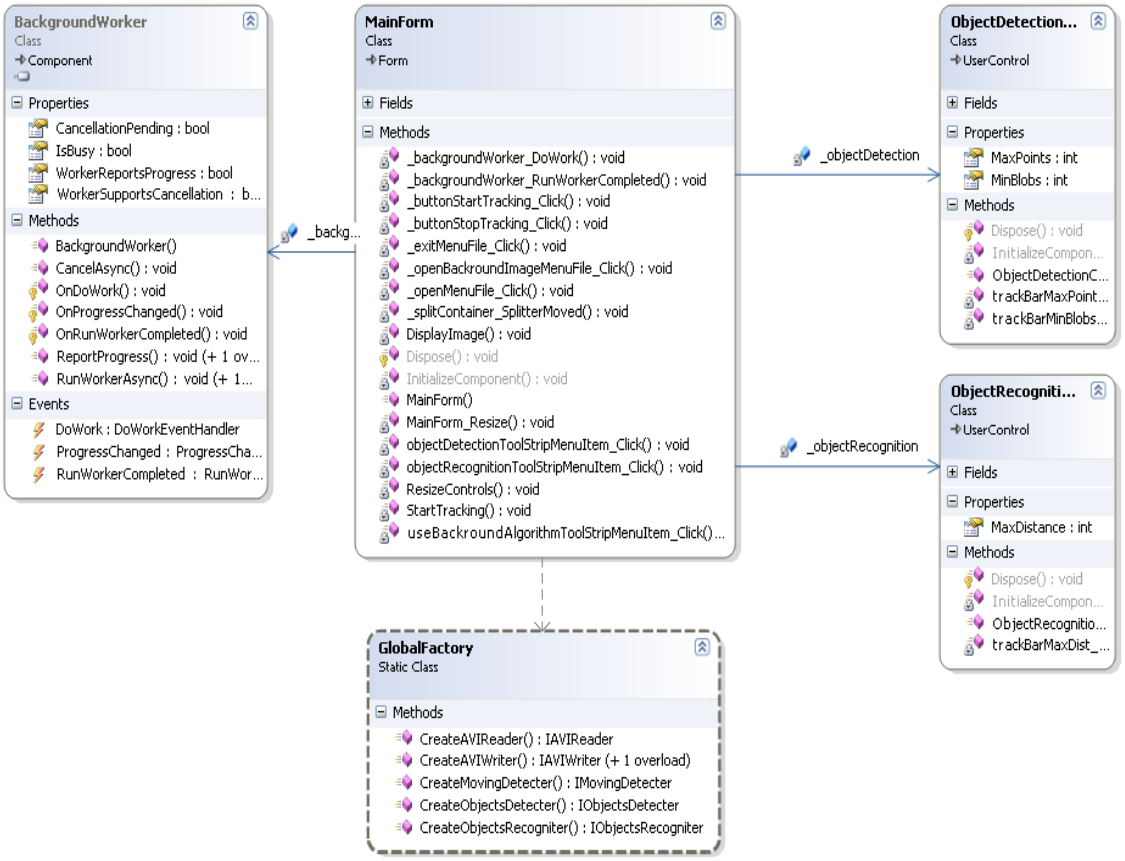

Рисунок 3.1 - Диаграмма классов модуля пользовательского интерфейса

BackgroundWorker – Класс, реализующий фоновый поток, непосредственно в данном потоке выполняются все операции задачи работы с изображением.

ObjectDetectionControl – Класс, предоставляющий графические элементы для настройки алгоритма выделения объектов. Данный класс реализован в виде элемента управления и может использоваться отдельно от данного модуля.

MainForm – Класс главного окна приложения, предоставляет все необходимые функции для работы с программой классификации, такие как открытие файла изображения, его отображение, настройку основных алгоритмов обеспечивающих сегментации изображения, средств управления программой и отображение результата классификации.

ObjectRecognitionControl – Класс, предоставляющий графические элементы для настройки алгоритма соотнесения объектов. Данный класс реализован в виде элемента управления и может использоваться отдельно от данного модуля.

GlobalFactory – Статический класс, реализующий паттерн абстрактная фабрика. Предназначен для предоставления интерфейсов на различные объекты.

3.4 Реализация модуля определения цвета

Реализация модуля определения цвета заключается в реализации алгоритма сегментации. Реализуемый алгоритм должен давать возможность работать в цветном пространстве RGB и в градациях серого HSV RGB (аббревиатура английских слов Red, Green, Blue — красный, зелёный, синий) — аддитивная цветовая модель, как правило, описывающая способ синтеза цвета для цветовоспроизведения. Аддитивной она называется потому, что цвета получаются путём добавления к черному.

HSV (англ. *Hue, Saturation, Value — тон, насыщенность, значение*) или HSB (англ. *Hue, Saturation, Brightness — тон, насыщенность, яркость*) — цветовая модель, в которой координатами цвета являются:

- Hue — цветовой тон, (например, красный, зелёный или сине-голубой). Варьируется в пределах 0—360°, однако иногда приводится к диапазону 0—100 или 0—1. В Windows весь цветовой спектр делится на 240 оттенков (что можно

наблюдать в редакторе палитры MS Paint), то есть здесь "Hue" приводится к диапазону 0-240 (оттенок 240 отсутствует, так как он дублировал бы 0).

– Saturation — насыщенность. Варьируется в пределах 0—100 или 0—1. Чем больше этот параметр, тем «чище» цвет, поэтому этот параметр иногда называют чистотой цвета. А чем ближе этот параметр к нулю, тем ближе цвет к нейтральному серому.

– Value (значение цвета) или Brightness — яркость. Также задаётся в пределах 0—100 или 0—1.

Диаграмма классов реализованного модуля определения движения представлена на рисунке 3.2.

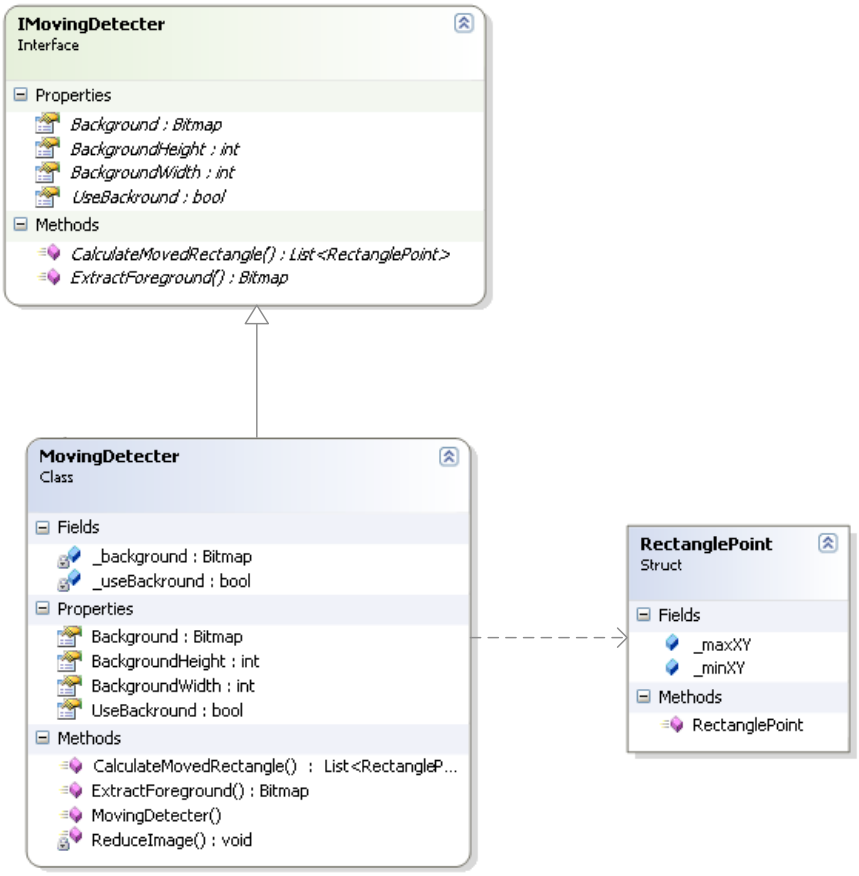

Рисунок 3.2 – Модуль определения цвета

IMovingDetecter - Интерфейс, декларирующий функции необходимые для определения цвета.

MovingDetecter - Класс реализующий интерфейс IMovingDetecter, позволяет получать изображение в разных цветовыз пространствах.

RectanglePoint – Структура, описывающая необходимый цвет.

В качестве алгоритма сегментации выбрана пирамидальная сегментация изображения.

Исходный код алгоритма сегментации представлен в приложении А.

3.5 Реализация модуля выделения объекта

Модуль выделения объектов строится на основе алгоритма определения цвета, который был описан в предыдущем разделе. Задача данного модуля на основании цветового пространства полученного изображения определить области с необходимым цветом и отделить их от основного изображения. Диаграмма классов модуля выделения объектов представлена на рисунке 3.3.

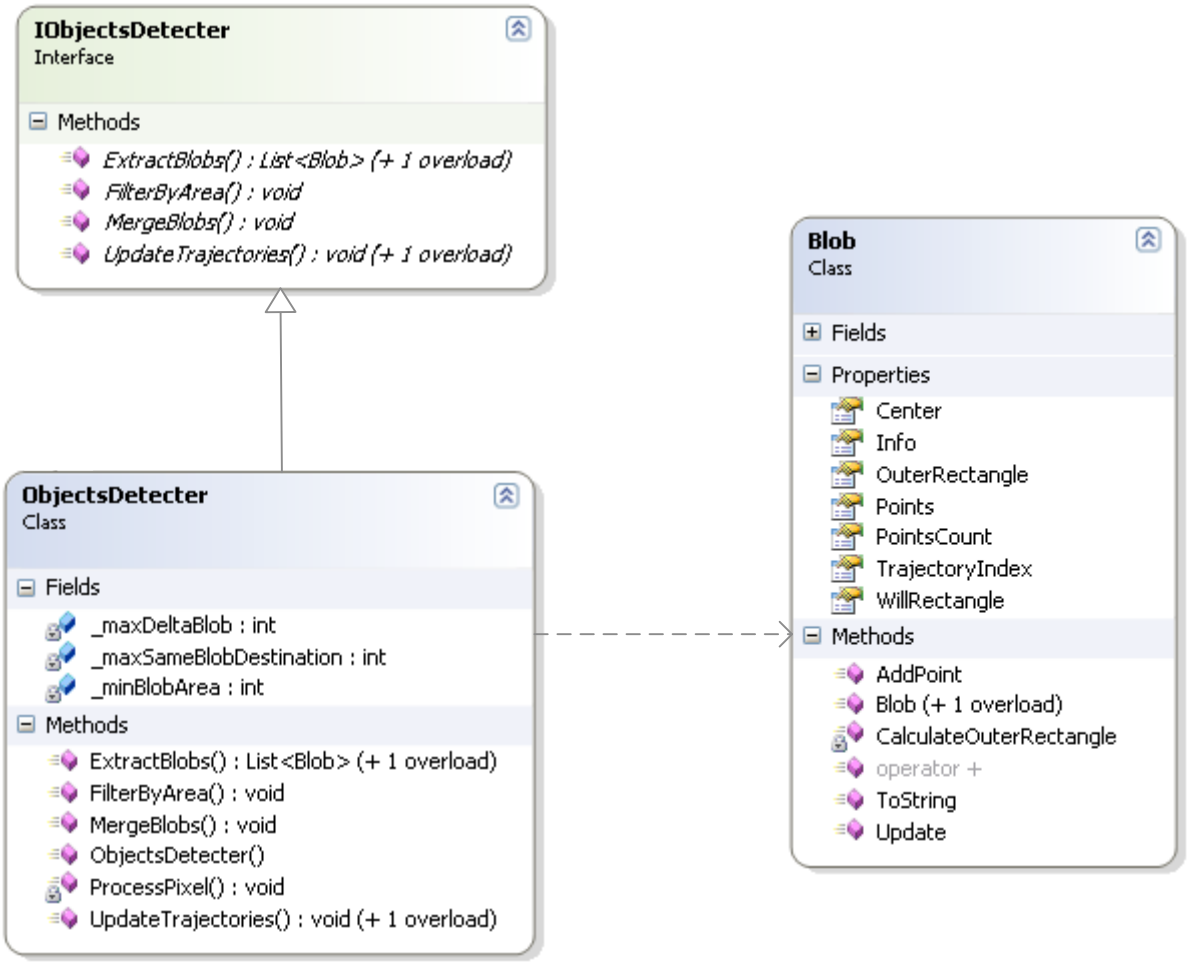

Рисунок 3.3 – Модуль выделения объекта

IObjectsDetecter - Интерфейс, декларирующий функции необходимые для выделения объектов.

ObjectsDetecter - Класс реализующий интерфейс *IObjectsDetecter*, позволяет получать из битовой маски массив объектов.

Blob - Класс описывает выделенный объект. Данный класс хранит информацию о пикселях, которые принадлежат данному объекту, геометрические координаты объекта и характеристические признаки объекта.

Исходный код алгоритма выделения объекта приведён в приложении Б.

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

4.1 Основные модули

Для выполнения поставленных задач программное средство должно обладать необходимым набором программных модулей, отвечающих за выполнение определенных операций. Основными модулями программы являются: выделение объекта и сегментация изображения . Далее рассмотрим каждый модуль в отдельности .

4.2 Модуль выделения объекта

Модуль выделения объекта предназначен для определения границы необходимого объекта на фоне всего изображения. Задача выделения сводится к задаче определения контуров. Контурный анализ — это один из важных и очень полезных методов описания, хранения, распознавания, сравнения и поиска графических образов/объектов. Алгоритм выделения объекта представлена на схеме ГУИР.400201.002 Э6

Контур — это внешние очертания (обвод) предмета/объекта. При проведении контурного анализа полагается, что контур содержит достаточную информацию о форме объекта и внутренние точки объекта во внимание не принимаются. Для определения контуров используется цепной код Фримена, принцип работы которого показан на рисунке 4.1. Цепные коды применяются для представления границы в виде последовательности отрезков прямых линий определённой длины и направления. В основе этого представления лежит 4- или 8- связная решётка. Длина каждого отрезка определяется разрешением решётки, а направления задаются выбранным кодом. Для представления всех направлений в 4-связной решётке достаточно 2-х бит, а для 8-связной решётки цепного кода требуется 3 бита. Библиотека OpenCV реализует удобные методы для детектирования и манипуляции с контурами изображения. Для поиска контуров используется функция *cvFindContours().* Функция имеет вид:

CVAPI(int) cvFindContours(CvArr* image, CvMemStorage* storage, CvSeq** first_contour,

int header_size CV_DEFAULT(sizeof(CvContour)),
int mode CV_DEFAULT(CV_RETR_LIST),
int method CV_DEFAULT(CV_CHAIN_APPROX_SIMPLE),
CvPoint offset CV_DEFAULT(cvPoint(0,0)));

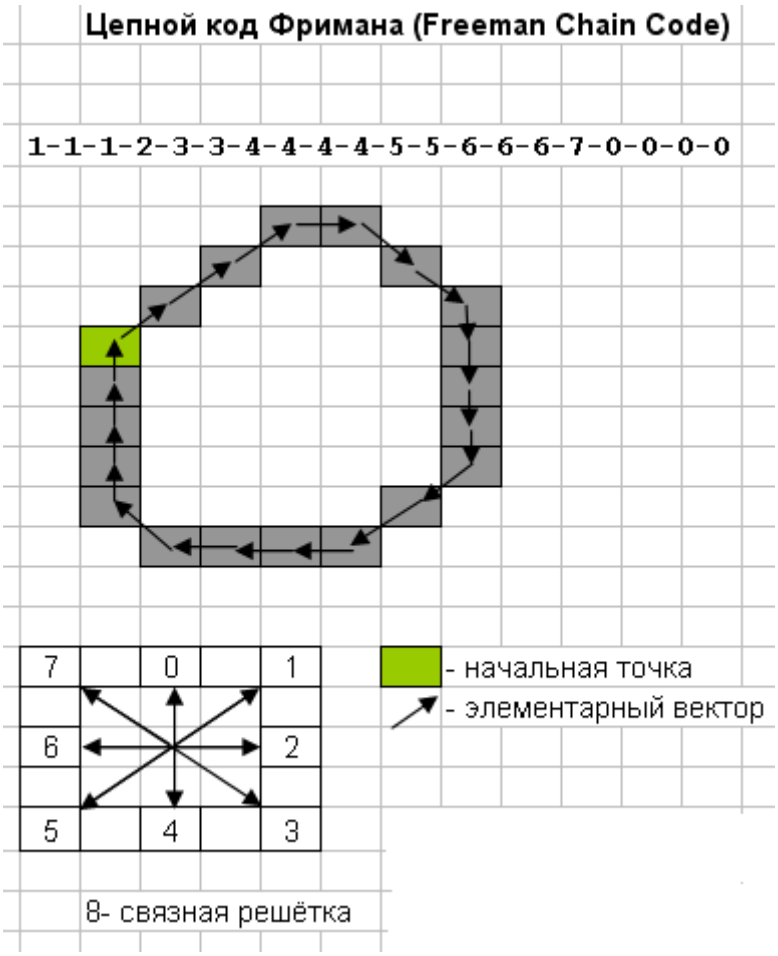

Рисунок 4.1 – Принцип работы цепного кода Фримена

Параметры функции следующие:

- image — исходное 8-битное одноканальное изображение (ненулевые пиксели обрабатываются как 1, а нулевые — 0);
- storage — хранилище памяти для хранения данных найденных контуров;
- header_size — размер заголовка элемента последовательности;
- mode — режим поиска.

В качестве основного алгоритма выделения объекта используется метод водораздела. В сегментации методом водораздела рассматривается абсолютная величина градиента изображения как топографической поверхности. Пиксели, имеющие наибольшую абсолютную величину градиента яркости,

соответствуют линиям водораздела, которые представляют границы областей. Вода, помещённая на любой пиксель внутри общей линии водораздела, течёт вниз к общему локальному минимуму яркости. Пиксели, от которых вода стекается к общему минимуму, образуют водосбор, который представляет сегмент. Преобразование водораздела вычисляет водосборные бассейны и линии хребтов, при том что водосборные бассейны - соответствующие области изображения, а линии хребтов – это границы этих областей. На рисунке 4.2 и 4.3 показаны исходное изображение и результат работы алгоритма водораздела.

Рисунок 4.2 – Исходное изображение

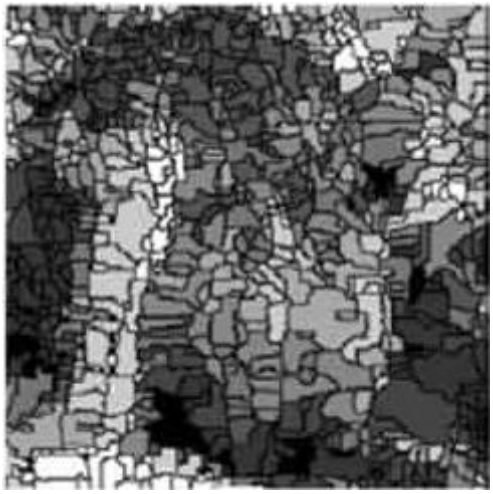

Рисунок 4.3 – Результат работы алгоритма водораздела

Основной проблемой данного алгоритма является чрезмерная сегментация, поскольку все границы и шумы отображаются в градиенте, что делает необходимым процесс удаления шума. Первый этап удаления шума в

начальном изображении состоит в применении морфологических операций закрытия/раскрытия, затем вычисляется морфологический градиент изображения без шума и выполняется нелинейное преобразование для уровней серого на градиенте изображения при помощи принципа Вебера, последний этап – вычисление водораздела по нелинейному, разбитому на области, градиентному изображению.

В ходе испытания алгоритма водораздела в качестве модуля выделения объекта он дал хорошие результаты в решении поставленной задачи, поэтому он используется в качестве алгоритма нахождения контуров.

Пример нахождения контуров представлен в Приложении А.

4.3 Модуль сегментации

Для сегментации, или выделения изображения используется алгоритм пирамидальной сегментации. Основной задачей пирамидальной сегментации является объединение соседних элементов, имеющих близкие признаки и не разделенных контуром. Эта процедура требует прослеживания контурных линий на всех уровнях пирамиды. Считается, что два вертикально или горизонтально соседствующих элемента разделены контуром в том случае, если расстояние между их отображениями в пространстве признаков. Задачей стадии первичной сегментации является преобразование исходного векторного изображения в первоначальный набор кластеров небольшого размера, в котором каждый сформированный кластер характеризуется как собственными параметрами, так и параметрами связи с соседними кластерами.

Основной задачей пирамидального этапа сегментации является объединение соседних элементов, имеющих близкие признаки и не разделенных контуром. Эта процедура требует прослеживания контурных линий на всех уровнях пирамиды. Считается, что два вертикально или горизонтально соседствующих элемента разделены контуром в том случае, если расстояние между их отображениями в пространстве объединенной текстурно-цветовой метрики) превышает некоторый заданный порог. Для второго и последующих уровней процедура обнаружения контуров также учитывает наличие контура на предыдущем уровне.

Алгоритм пирамидальной сегментации показан на схеме ГУИР.400201.003 ПД.

Принцип пирамидальной сегментации показан на рисунке 4.2.

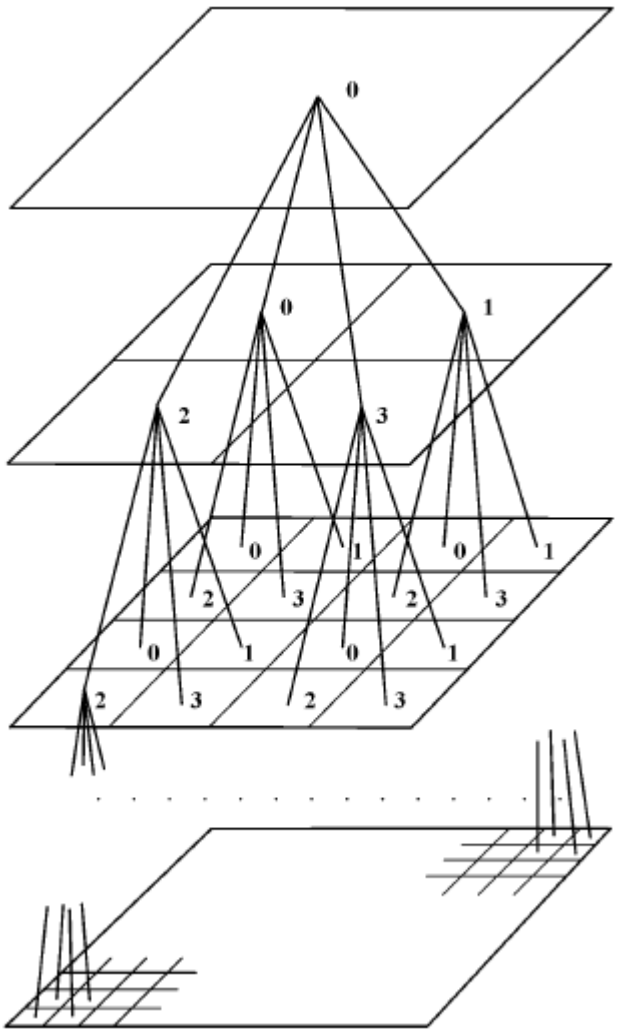

Рисунок 4.2 – Принцип пирамидальной сегментации

На каждом шаге на основе анализа четырех нижних узлов уровня n создается узел уровня n + 1, а в соответствующей структуре нового узла запоминается информация об узлах предыдущего уровня, соединенных с данным узлом, их средней яркости, наличии контуров. Таким образом, каждый узел является вершиной некоторого квадродерева, охватывающего расположенные под ним элементы изображения, и содержит информацию о поддеревьях предыдущего уровня.

Применяется прямой проход вверх при сегментации. То есть, при этом происходит рекурсивный анализ всех уровней пирамиды, начиная с самого нижнего (исходного изображения) и заканчивая верхним уровнем, состоящим из одного узла; одновременно с этим строится само квадродерево.

Для сокращение числа сегментов при условии минимальной потери (т.е. слияния разнородных) объектов было решено реализовать алгоритм вторичной сегментации. Она сводится к упрощению графа кластеров,

построенного во время первичной сегментации. Это достигается сравнением признаков пар соседних узлов и слиянием тех из них, признаки которых близки. Для сравнения признаков кластеров используется мера расстояния между ними. Если расстояние между парой соседних узлов меньше некоторого задаваемого порога, то эти узлы сливаются, а их признаки усредняются. Признаки новых кластеров, получаемых в результате слияния, формируются как усреднение признаков объединяемых кластеров или элементов. Тем самым они могут быть отображены в то же пространство признаков, а значит, допустимо использовать ту же самую метрик и для измерения межкластерного расстояния. Для достижения лучших результатов было добавлено расширение пространство признаков, а в качестве основных характеристик были добавлены: размер кластера (число элементов в кластере) и интегральный текстурный уровень. В расширенном пространстве признаков расстояние между кластерами u и v задается следующей формулой (4.1):

$$\widetilde{D}(u; v) = [\widetilde{w}_c + \widetilde{w}_t(1 - t(u; v)p(u; v))]s(u; v)D_c(u; v) + \\ + [(1 - s(u; v))(\widetilde{w}_c + \widetilde{w}_t) + \widetilde{w}_t s(u; v)t(u; v)p(u; v)]D_t(u; v). \quad (4.1)$$

Здесь $D_c(u; v)$ и $D_t(u; v)$. — цветовая и текстурная части в пространстве признаков, задаваемые с той лишь разницей, что вместо признаков отдельных элементов подставляются соответствующие признаки кластеров.. Функции p, t и s зависят от размеров, уровня текстуры и цветовой насыщенности кластеров u и v.

Далее ввелась процедура вторичной сегментации. Процедура вторичной сегментации построена по следующей итеративной схеме. Набор узлов графа (кластеров) анализируется в порядке убывания размеров кластеров. Для каждого очередного узла, который в данном случае выступает в роли базового узла (базового кластера), просматривается набор соединенных с ним узлов (соседних кластеров). Если согласно расстояние между базовым узлом и очередным соседним не превышает заданного порога, то этот (соседний) узел сливается с базовым. Это означает, что все связи соседнего узла переориентируются на базовый узел, а также корректируются все связи и характеристики объединенного узла (кластера). Затем операция сравнения повторяется со следующим соседним узлом. По окончании списка соседних узлов осуществляется переход к следующему базовом узлу. Процедура заканчивается, когда обработаны все узлы графа.

Разработанный алгоритм продемонстрировал удовлетворительные результаты на большинстве из доступных изображений, среди которых:

тестовые изображения сцен ограниченной сложности, реальные сцены и портреты, изображения из различных баз данных и библиотек изображения, составленные из текстур альбома. Пример кода вторичной сегментации показан в Приложении Б.

Для реализации в программе используется функция cvPyrSegmentation. Пример использования функции пирамидальной сегментации в коде показан в Приложении В.

Функция имеет параметры:

- storage – сохранение результирующей последовательности;
- comp – указатель на выходную последовательность сегментированных компонентов.
- level – максимальный уровень “пирамиды” для сегментации;
- threshold1 – порог для установления связи;
- threshold2 – порог для кластеризации сегментов.

5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

5.1 Общие сведения

Программное средство представляет собой законченный продукт со своим интерфейсом, а так же необходимым набором настроек. Внешний вид программы классификации представлен на рисунке 5.1.

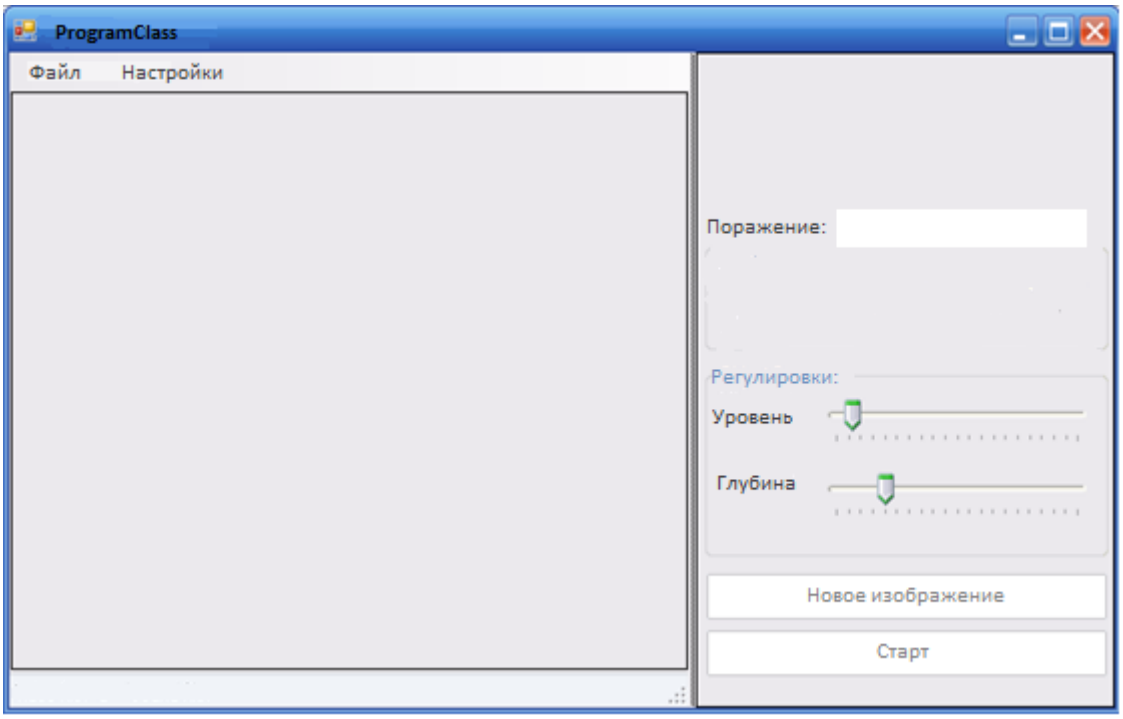

Рисунок 5.1 – Внешний вид программы

В программе имеются необходимые регулировки для подстройки качества изображения.

Методика испытаний сводится к проверке получения удовлетворительного результата работы программного средства. Для этого будут отработаны основные этапы работы с программой с фиксацией результата работы после каждой операции: выбор файла изображения, его загрузка и отображение, выполнения основного назначения и проверка результата работы, проверка на ошибку (в данном случае – загрузка изображения здоровой растительности).

5.2 Испытание программного средства

Испытание подразумевает под собой проверку на выполнение основных функций, ошибочных значений.

Испытание начинаем проводить с загрузки изображения. В качестве исходного файла изображения может выступать картинка с расширением .jpg неограниченного размера (в целях быстродействия рекомендуется использовать файлы размером до 3-х мегабайт) Для этого выбираем меню «Файл» и выбираем «Открыть изображение». В появившемся диалоговом окне открываем необходимый файл. Как видно из рисунка 5.2, данная операция выполняется.

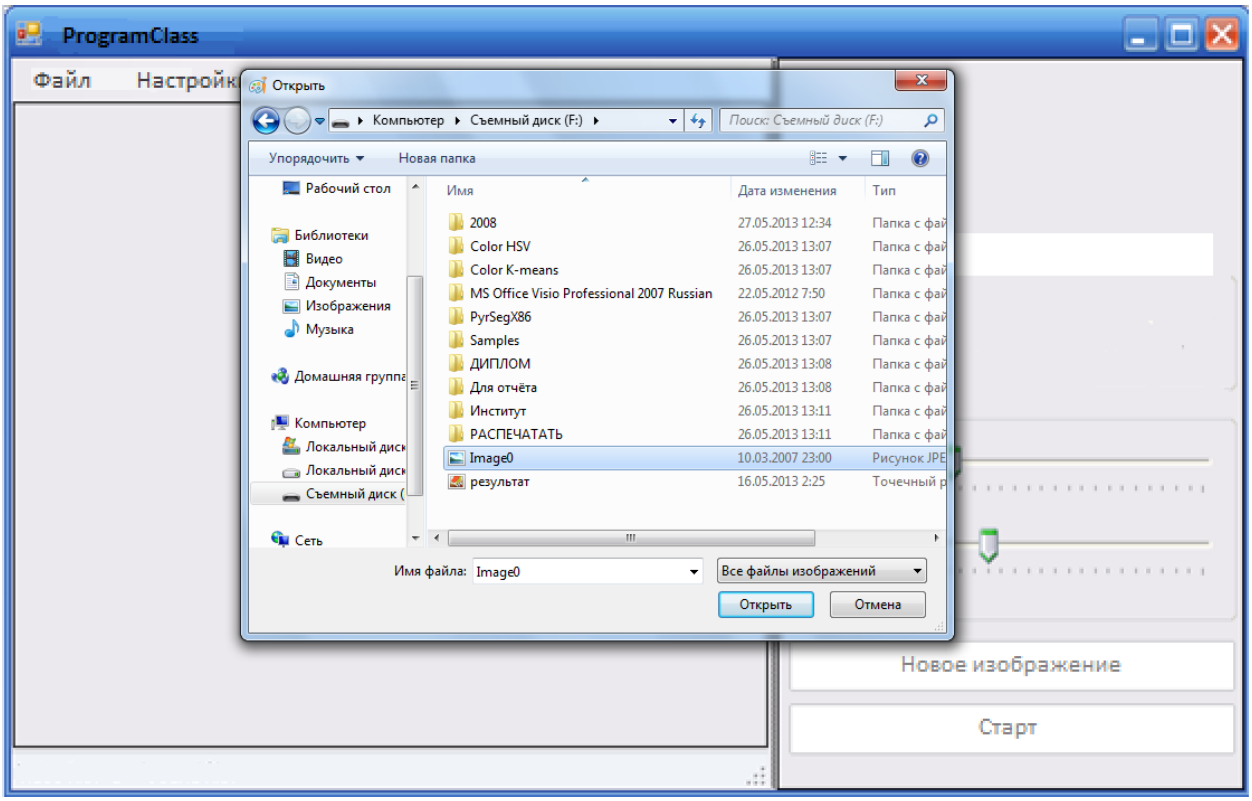

Рисунок 5.2 – Диалоговое окно выбора изображения

Далее, после выбора, необходимое изображение появляется на рабочей области программы, как это видно на рисунке 5.3.

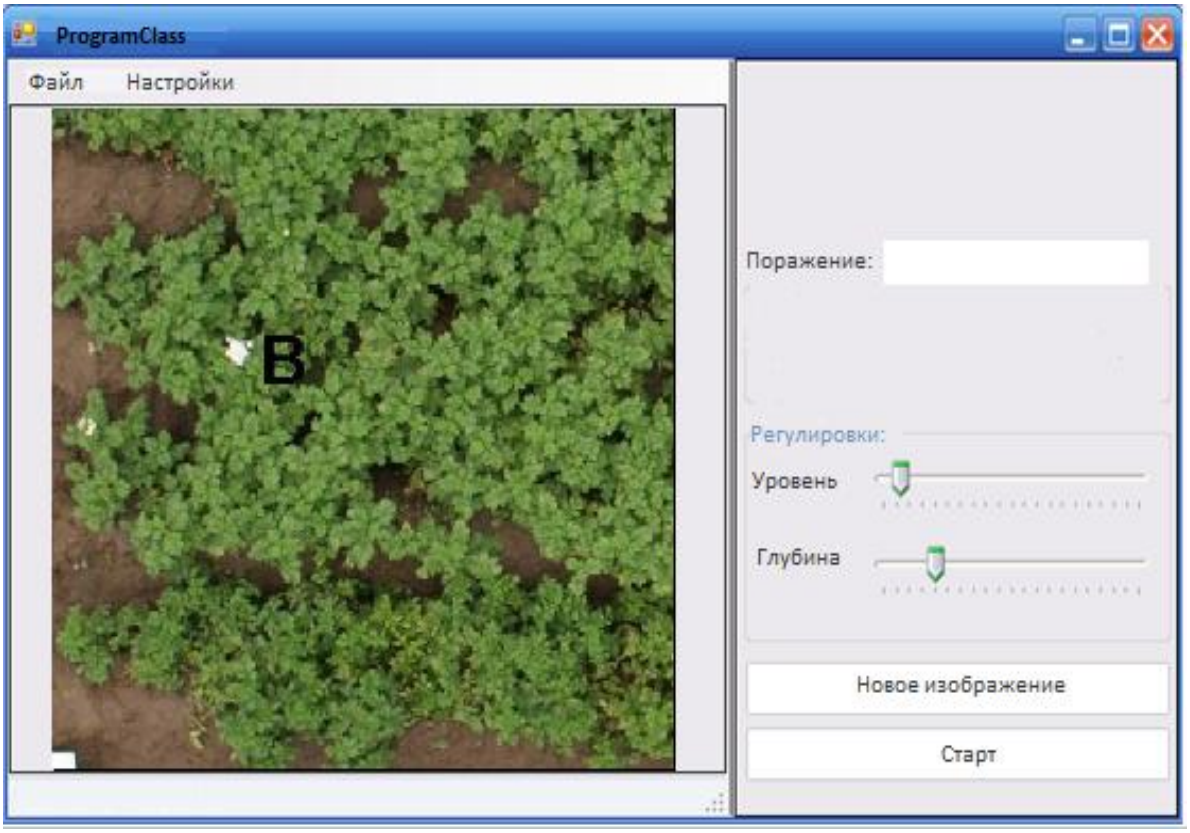

Рисунок 5.3 – Исходное изображение

После нажатия кнопки “Старт” запустится процесс обработки изображения и классификации. Спустя время на экране появится результат работы программы с изображением участка с выделенными поражениями и выдан результат о типе заболевания. На исходном изображении можно заметить поражённые участки. Это показано на рисунке 5.4, где обведена область с пораженной растительностью. Это же изображение будет использоваться и для проверки на параметрах сегментации, задаваемых ползунками «Уровень» и «Глубина». В ходе работы с изображением программа должна провести несколько этапов обработки и выдать результат на экран с указанием типа поражения. Это и есть основная задача программного средства классификации изображений для мониторинга состояния сельскохозяйственной растительности – определение и классификация поражения по данным исходного изображения. Успешное выполнение этой операции определяет качество выполнения разработки программного средства и его пригодность для использования и конкуренции с другими продуктами.

Рисунок 5.4 – Изображения с поражёнными участками

После обработки изображения эти области будут выделены на общем фоне и приведена классификация, как показано на рисунке 5.5. Как видно из рисунка на полученном изображении выделены участки с наибольшим поражением. Они имеют яркий окрас для лучшего зрительного восприятия. Так же в графе «Поражения» появляется информация о текущем заболевании. Успешная отработка основных задач говорит о правильно выбранных функциях и разработки алгоритмов в конкретной задаче – определении поражённых участков на изображении растительности и её классификация.

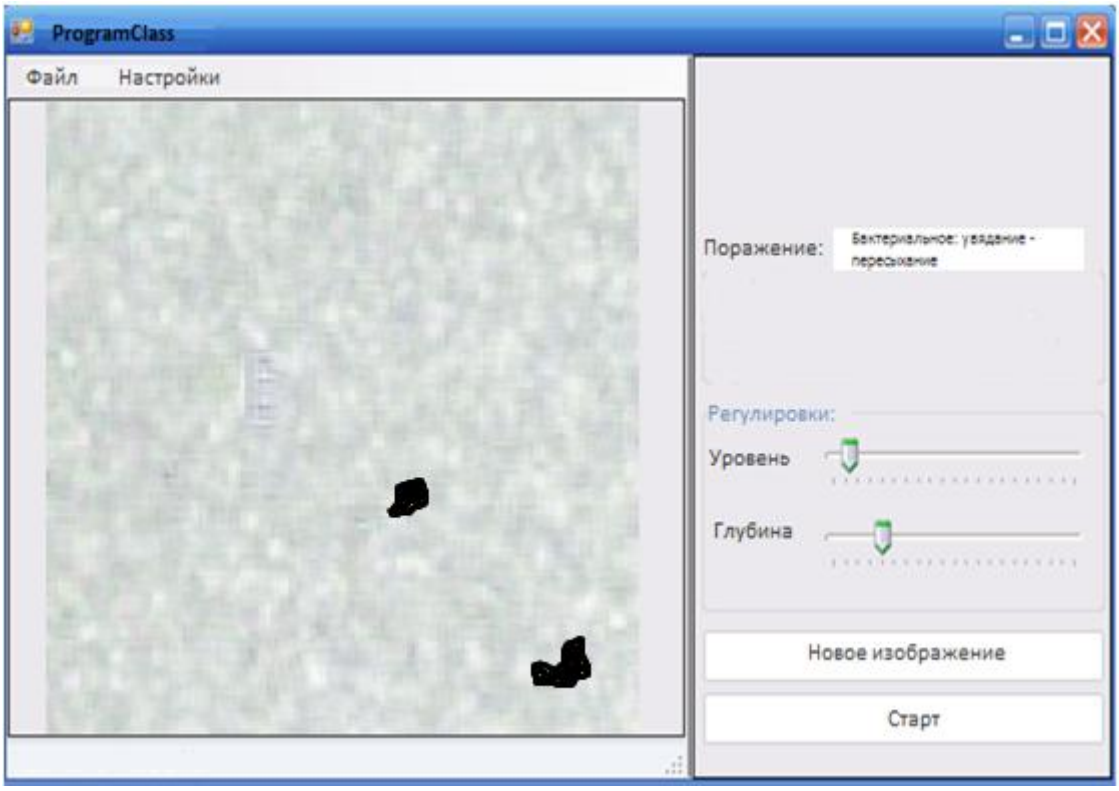

Рисунок 5.5 – Результат работы программы

Если в процессе работы программы поменять значения уровня и глубины, то результаты преобразования получатся менее точными. Как видно из рисунка 5.6, при перемещении ползунков в большую сторону точность показаний снижается в данном случае. В процессе эксплуатации можно подбирать положения ползунков для достижения лучших результатов. Однако, рекомендуется использовать высокие значения уровня и глубины при более крупном масштабе, то есть при изображении единичного растения, а при исследовании группы растений – низкие значения глубины и уровня. Это объясняется более точном сегментировании с высоким значением числа пирамиды (уровень) и глубины проработки.

Далее загружаем в программу изображение со здоровой растительностью. На рисунке 5.7 представлена работа программы с изображением без поражения. Для картинки с более крупным масштабом оставляем значения уровня и глубины на уровне, ближе к максимальному. На рисунке 5.8 показан результат работы программы без поражений.

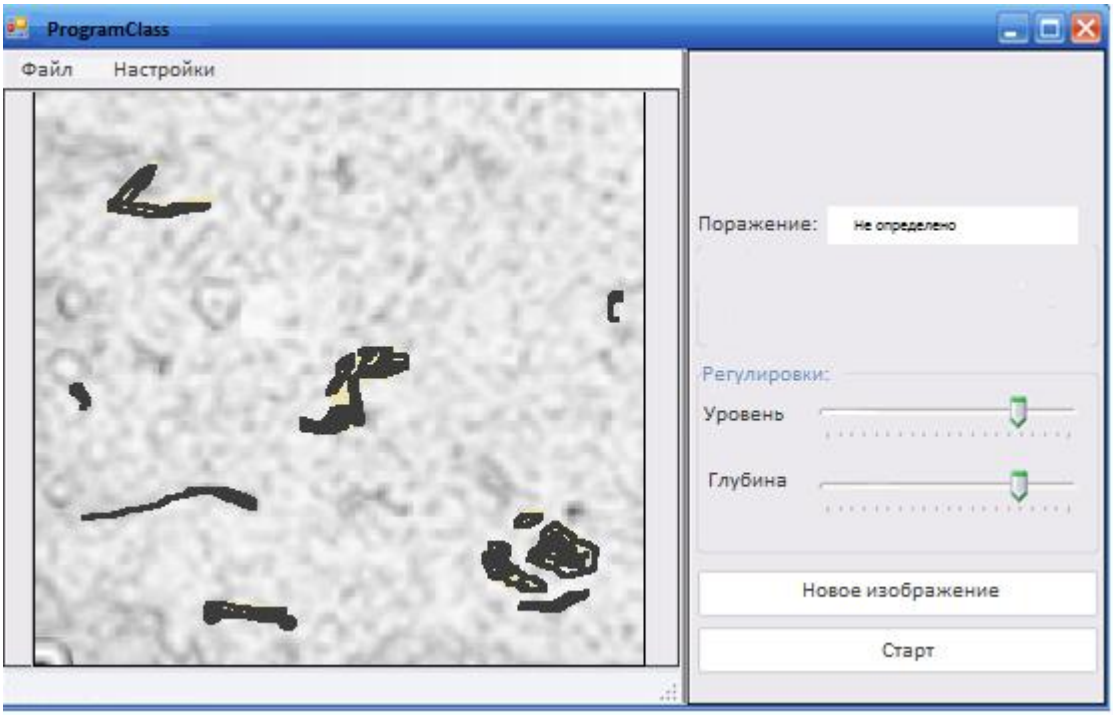

Рисунок 5.6 – Результат с изменённым уровнем и глубиной

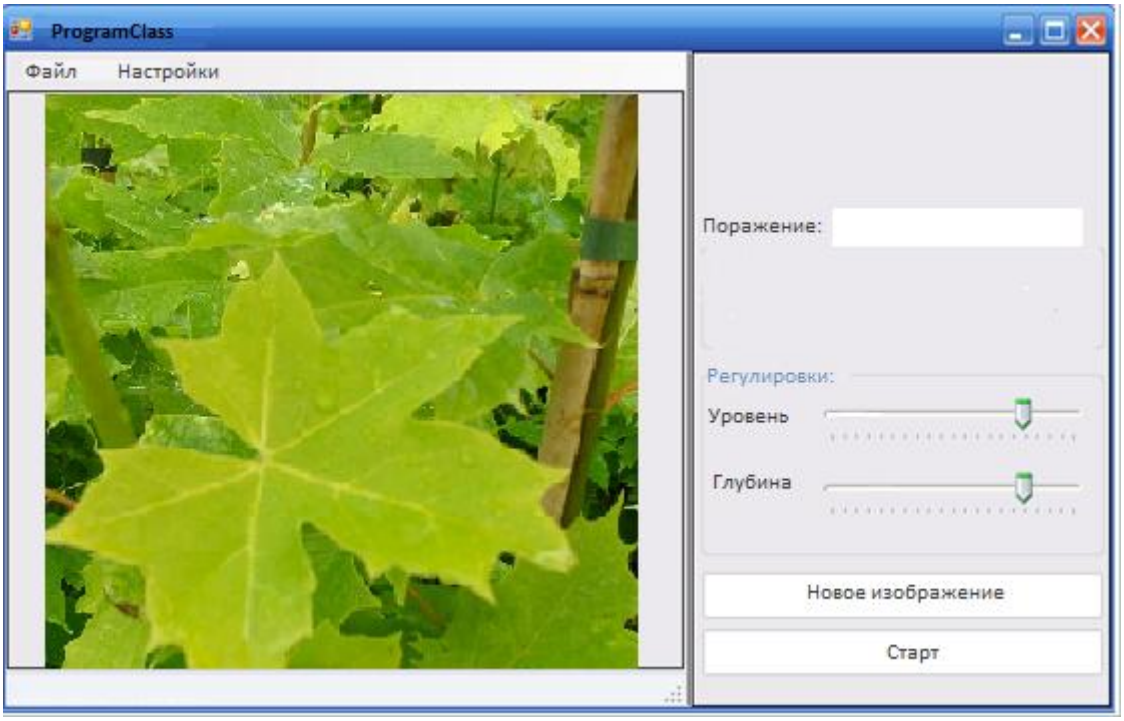

Рисунок 5.7 – Загрузка изображения здоровой растительности

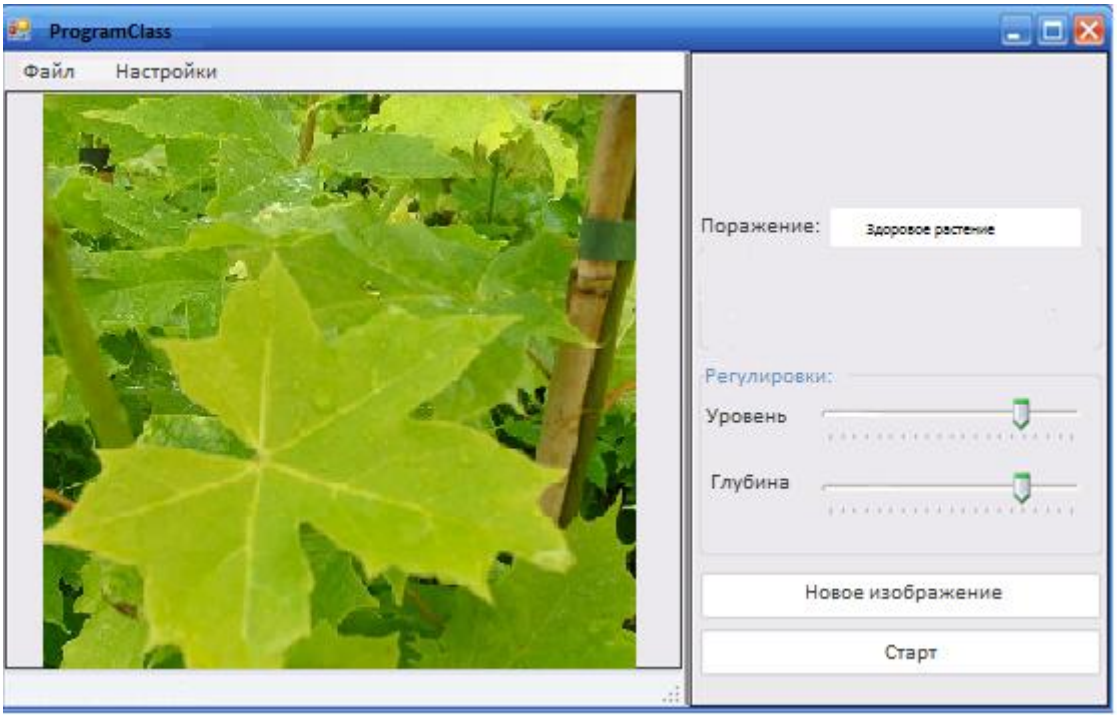

Рисунок 5.8 – Результат работы со здоровым растением

Как видно из рисунка 5.8, изображение загружается без изменений, а в графе поражений указывается, что растение здорово.

Таким образом, в ходе тестирования программного средства классификации поражений сельскохозяйственной растительности не было замечено аварийных ситуаций, программа справилась с поставленной задачей выделения областей и классификации поражений. Есть настройка для более точной подстройки параметров при различных входных изображениях.

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

6.1 Требования к аппаратной составляющей

В этом разделе будут описаны системные требования, предъявляемые к ЭВМ, на которой будет запускаться данное программное средство и программные требования.

Для запуска программы необходима ЭВМ со следующими характеристиками:

- операционная система Windows Seven;
- процессор с тактовой частотой не ниже 1.5 Гц;
- оперативная память не меньше 1 Гб;
- объём дискового пространства для программы со всеми модулями 3 Гб.

Такие требования диктуются наличием библиотеки OpenCV, а так же необходимостью быстрой обработки изображения.

Программное средство было испытано на ЭВМ, работающей на операционной системе Windows Seven, процессоре с тактовой частотой 1.8 Гц, объёмом оперативной памяти 2.5 Гб, свободное место на жёстком диске 10 Гб, из них всей системой занято 3 Гб. При таких характеристиках никаких проблем с эксплуатацией программного средства не возникло. Такими требованиями обладают большинство современных ЭВМ.

6.2 Требования к программной составляющей

Для реализации работы программного средства в первую очередь необходима библиотека OpenCV. OpenCV (англ. *Open Source Computer Vision Library*, библиотека компьютерного зрения с открытым исходным кодом) — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Она содержит все необходимые для работы библиотеки. В процессе написания дипломного проекта была использована библиотека OpenCV 2.4.5. На данный момент это последняя версия библиотеки и она распространяется бесплатно на условиях лицензии BSD. Папку с библиотекой необходимо помещать в корень диска C.

Также для функционирования программы необходимо в папку system32 положить файл tbb_debug.dll. TBB - это кроссплатформенная библиотека шаблонов C++, разработанная компанией Intel для параллельного программирования и содержащая алгоритмы и структуры данных,

позволяющие избежать сложностей, возникающих при использовании традиционных реализаций потоков.

6.3 Основные элементы управления и работа с программой

Рабочее окно программы состоит из нескольких элементов. На рисунке 6.1 показан внешний вид окна программы.

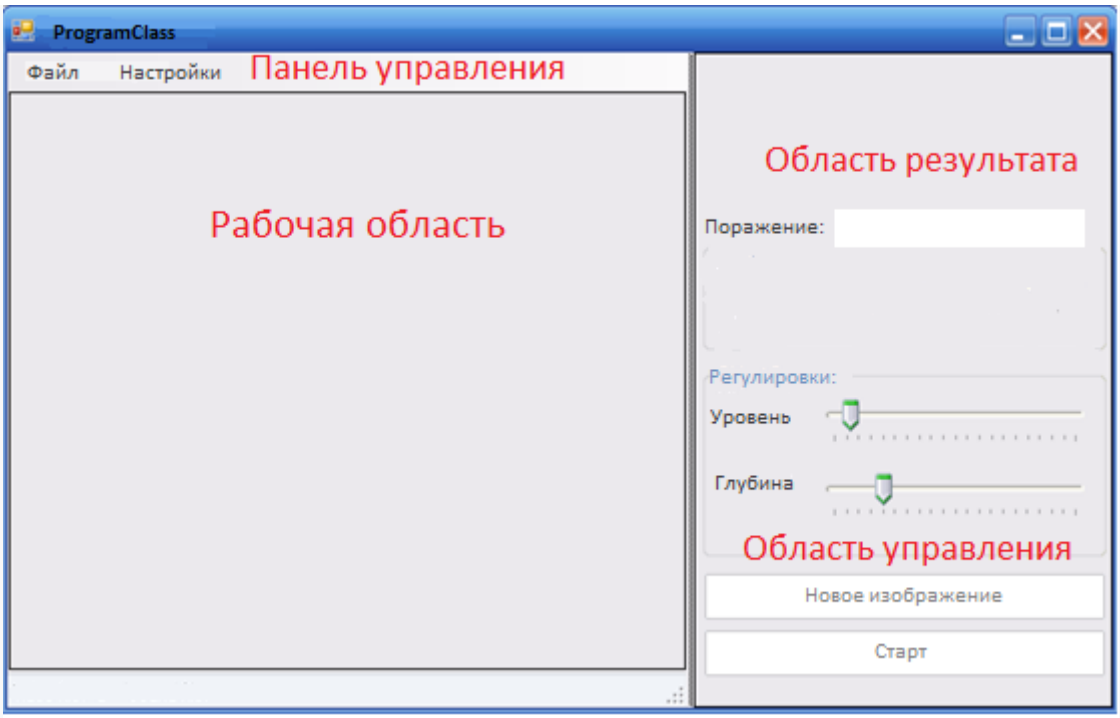

Рисунок 6.1 – Области рабочего окна

Ниже будут рассмотрены эти области в отдельности.
На рисунке 6.2 представлена два пункта рабочего меню.

Рисунок 6.1 – Пункты рабочего меню

В пункте «Файл» находятся элементы для работы с изображением: создать и сохранить. В пункте «Настройки» - опции изображения.

На рисунке 6.3 изображена рабочая область. Она предназначена для отображения входной и выходной информации – изображений

сельскохозяйственной продукции. При выборе изображения оно разворачивается на всю область.

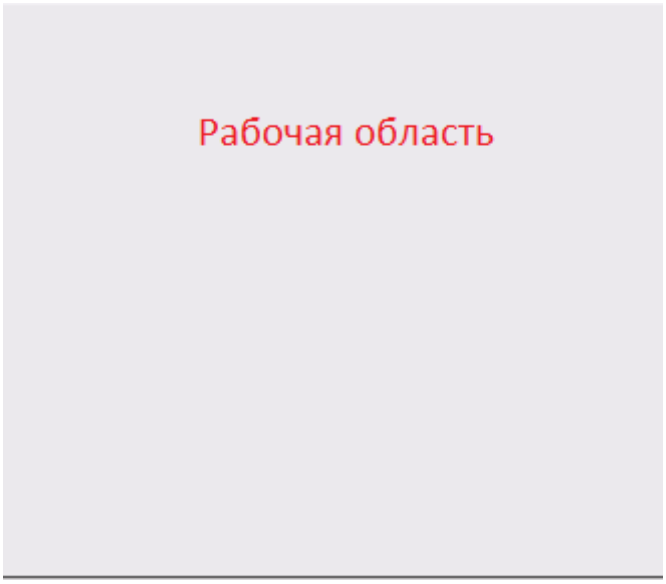

Рисунок 6.3 – Рабочая область программы

В правой нижней части рабочего окна находится Область управления, изображённая на рисунке 6.4.

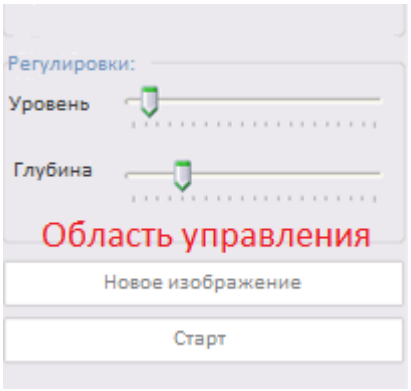

Рисунок 6.4 – Область управления

Она включает в себя кнопки начала работы и загрузки изображения. А так же имеются два ползунка для изменения параметров уровня и глубины. Перемещать ползунки необходимо курсором мыши, зажав ползунок левой клавишей и перемещать его в нужном направлении.

Чуть выше расположена Область результата. Она представлена небольшим полем, на котором отображается результат обработки изображения – поражение или здоровое растение. Данная область визуально изображена на рисунке 6.5.

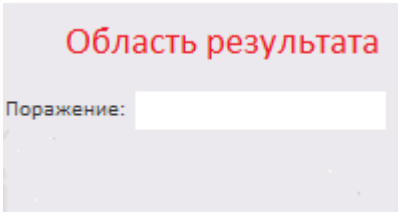

Рисунок 6.5 – Область результата

Так же в левом верхнем углу расположены стандартные элементы управления Windows-окном: свернуть, развернуть и закрыть. На рисунке 6.6 изображены стандартные элементы управления окном.

Рисунок 6.6 – Стандартное управление окном

Последовательность работы такова: при первом запуске необходимо в меню “Файл” выбрать пункт “Открыть” и из списка выбрать необходимое изображение. Далее нажать кнопку “Старт”, после чего на экране появится результат работы программы в виде изображения. При следующем загрузке изображения его можно выбрать из области управления, нажав кнопку “Новое изображение”. При необходимости, изменяют значения глубины и уровня путём перемещения курсором мыши соответствующих ползунков в области управления. По завершении работы необходимо закрыть программу кнопкой “Закрыть” (крестик) в панели управления окном.

Таким образом, данное программное средство обладает необходимым пользовательским интерфейсом, не перегруженным ненужными функциями и легко в освоении за счёт минимального количества элементов управления. Не маловажным фактором является и то, что сама программа не требует настройки, а способна выполнять свои функции после первого запуска перед соответствующей настройкой компьютера. Программа подойдёт для освоения и эксплуатации большинству пользователей ПК.

7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРИМЕНЕНИЯ РАЗРАБОТКИ ПРОГРАММНОЙ СИСТЕМЫ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ ДЛЯ МОНИТОРИНГА СОСТОЯНИЯ СЕЛЬСКОХОЗЯЙСТВЕННОЙ РАСТИТЕЛЬНОСТИ

7.1 Краткая характеристика программного средства

Программная система классификации изображений для мониторинга состояния сельскохозяйственной растительности, разрабатываемая в рамках данного дипломного проекта, предназначена для контроля за состоянием растительности, выявления и классификации поражений. Разработка данного программного продукта предусматривает исследование, анализ требований, проектирование и разработка, и относится ко третьей категории сложности и классифицируется, как функциональное ПС. Категория новизны – В.

7.2Определение объема и трудоемкости ПС

Базой для расчета плановой сметы затрат на разработку данного ПС является объем ПС. Для оценивания объема ПС в дипломном проекте в качестве единицы измерения используется строка исходного кода (LinesOfCode, LOC), которая представляет собой универсальную метрику для создания любых программных продуктов.

Стоимостная оценка программного средства и определение экономического эффекта у разработчика предполагают составление сметы затрат, которая в денежном выражении включает следующие статьи расходов:

- заработную плату исполнителей, основную ($З_о$) и дополнительную ($З_д$);
- отчисления в фонд социальной защиты населения ($З_{сз}$);
- налоги, от фонда оплаты труда ($Н_е$);
- материалы и комплектующие (М);
- спецоборудование ($Р_с$);
- машинное время ($Р_м$);

На основании информации о функциях разрабатываемого ПС, среды разработки определен номинальный и уточнённый объём функций и общий объем ПС, представленный в таблице 7.1. Уточнённый объём определяется с учётом условий разработки ПС.

Общий объем (V_o) программного продукта определяется исходя из количества и объема функций, реализуемых программой:

$$V_o = \sum_{i=1}^{n} V_i, \qquad (7.1)$$

где $V_i -$ уточнённый объём отдельной продукции ПС (LOC);

$n -$ количество учитываемых характеристик.

По объему ПС и нормативам затрат труда в расчете на единицу объема определяется нормативная и общая трудоемкость разработки ПС. Нормативная трудоемкость ($T_н$) определяется по таблице укрупненных норм времени на разработку ПО и при объеме V_o = 5940 LOC составляет$T_н$ = 135 чел./дн.

Таблица 7.1 - Характеристики функций и их объём

Номер функции	Содержание функции	Объём функции по каталогу	Объём функции уточнённый
101	Организация ввода информации	150	120
109	Организация ввода/вывода информации в интерактивном режиме	320	250
301	Формирование последовательного файла	290	230
305	Обработка файлов	720	620
309	Формирование файлов	1020	900
505	Управление внешней памятью	200	150
506	Обработка ошибочных и сбойных ситуаций	410	350
705	Формирование и вывод	3500	2900
707	Графический вывод результата	480	420
	Итого:	7020	5940

В таблице 7.2 представлены характеристики функций и их объём.

Обеспечение хранения, ведения и поиска данных в сложной структуре позволяет применить к объему ПС коэффициент$К_с$, рассчитанный по формуле (7.2):

$$К_с = 1 + \sum_{i=1}^{n} К_i = 1 + 0{,}07 = 1{,}07, \qquad (7.2)$$

где $К_с -$ коффициент, соответствующий степени повышения сложности ПС за счёт конкретной характеристики; n–количество учитываемых характеристик.

Степень использования в разрабатываемом программном средстве стандартных модулей определяется их удельным весом в общем объеме программы. Коэффициент, учитывающий степень использования при разработке стандартных модулей в данном дипломном проекте: $К_т = 0{,}7$.

Сравнение характеристик разрабатываемого программного средства с имеющимися аналогами позволяет определить экспертным путем степень его новизны. При установлении коэффициента новизны учитываются степень новизны программного средства и предназначение его для новых или освоенных типов ПК, для новых или освоенных ОС. Коэффициент новизны разрабатываемого ПС: $К_н = 0{,}7$.

По уточненному объему ПС и нормативам затрат труда в расчете на единицу объема определяются нормативная и общая трудоемкость разработки ПС.

Общая трудоёмкость представлена выражением (7.3):

$$Т_о = Т_н * К_т * К_с * К_н = 135 * 0{,}7 * 1{,}07 * 0{,}7 = 70 \frac{чел}{дней}. \qquad (7.3)$$

На основе общей трудоемкости определяются плановое число разработчиков ($Ч_р = 1$) и плановые сроки, необходимые для реализации проекта в целом ($Т_р$). При этом могут решаться следующие задачи:

- расчет числа исполнителей при заданных сроках разработки проекта;
- определение сроков разработки проекта при заданной численности исполнителей.

Так как специфика выполнения дипломных проектов подразумевает работу одного студента, то в данном случае при заданной численности исполнителей необходимо решить задачу определения сроков разработки.

Эффективный фонд времени одного работника – 236 дней. Срок разработки проекта ($Т_р$) определяется по формуле:

$$Т_р = \frac{Т_о}{Ч_р * Ф_{эф}} = \frac{70}{1 * 236} = 0{,}4\ г. \quad (7.4)$$

Таким образом, рассчитанный срок разработки проекта составит около пяти месяцев.

7.3 Расчет сметы затрат и цены заказного ПС

В случае разработки программного средства на заказ основной статьей расходов на его создание является заработная плата исполнителей проекта, в число которых принято включать инженеров-программистов, участвующих в написании кода, руководителей проекта, системных архитекторов, дизайнеров, разрабатывающих пользовательский интерфейс, разработчиков баз данных, Web-мастеров и других специалистов, необходимых для решения специальных задач в команде. Заработная плата руководителей организации и работников вспомогательных служб (инфраструктуры) учитывается в накладных расходах. Предположим, что разрабатываемое для дипломного проекта программное средство одновременно является коммерческим заказом.

Должность руководителя дипломного проекта будет рассматриваться как должность руководителя коммерческого проекта. Студент-дипломник – как программист-исполнитель. Таким образом, на разработке проекта будут заняты:

- Начальник отдела – тарифный разряд – 16; тарифный коэффициент – 3,72; плановый фонд рабочего времени – 20 дней.

- Программист II категории – тарифный разряд – 12; тарифный коэффициент – 2,84; плановый фонд рабочего времени – 100 дней.

При определении заработной платы будет использован коэффициент премирования – 1,6 и тарифная ставка первого разряда – 250 000 белорусских рублей от 01.04.2012 года.

Расчет часовой тарифной ставки начальника отдела и программиста вычисляется формулами (7.5) и (7.6):

$$Т_{ч1} = \frac{Т_{м1} * Т_к}{Ф_р} = \frac{250000 * 3{,}72}{170} = 5470{,}59\ руб., \quad (7.5)$$

где $Т_{м1}$ - месячная тарифная ставка первого разряда;

$Т_к$ – тарифный коэффициент;

$Ф_р$ – установленная при 40-часовой неделе норма рабочего времени.

$$Т_{ч2} = \frac{Т_{м2} * Т_к}{Ф_р} = \frac{250000 * 2{,}84}{170} = 4176{,}47\ руб. \quad (7.6)$$

Расчёт основной заработной платы выполняется по формуле (7.7):

$$З_о = \sum_{i=1}^{n} Т_{чi} * Т_ч * Ф_п * К = (5470{,}59 * 8 * 20 * 1{,}7) + (4176{,}47 * 8 * * 100 * 1{,}7) = 7167999{,}68 \text{ руб.}, \quad (7.7)$$

где $Т_{чi}$ – часовая тарифная ставка i-го исполнителя;

$Т_ч$ – количество часов работы в день;

$Ф_п$ – плановый фонд рабочего времени i-го исполнителя;

К – коэффициент премирования.

Дополнительная заработная плата включает выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов и других выплат, не связанных с основной деятельностью исполнителей), и определяется по нормативу в процентах к основной заработной плате:

$$З_д = \frac{З_о * Н_д}{100\%} = \frac{7167999{,}68 * 10\%}{100\%} = 716799{,}9 \text{ руб.}, \quad (7.8)$$

где $Н_д$ (10%) – норматив дополнительной заработной платы.

Расчет отчислений в фонд социальной защиты населения:

$$З_{сз} = \frac{(З_о + Н_д) * Н_{сз}}{100\%} = \frac{(7167999{,}68 + 716799{,}9) * 34}{100\%} = = 2680831{,}8 \text{ руб.}, \quad (7.9)$$

где $Н_{сз}$ – норматив отчислений в фонд социальной защиты населения,

$Н_{сз}$= 34%

Расчет налогов от фонда оплаты труда, уплачиваемых единым платежом:

$$Н_е = \frac{(З_о + Н_д) * Н_{не}}{100\%} = \frac{(7167999{,}68 + 716799{,}9) * 4}{100\%} =$$

=57343 руб., (7.10)

где $Н_{не}$ – норматив налога уплачиваемого единым платежом, $Н_{не}$ = 4%.

Расходы по статье “Материалы” отражают расходы на съемные носители, бумагу, красящие ленты и другие материалы, необходимые для разработки программного средства. Нормы расхода материалов для данного проекта будет рассчитана по нормативу к фонду основной заработной платы разработчиков. Сумма затрат на расходные материалы рассчитывается по формуле:

$$M = \frac{З_м * З_о}{100\%} = \frac{5\% * 7167999{,}68}{100\%} = 358399{,}98 \text{ руб.}, \quad (7.11)$$

где $Н_м$ – норма расхода материалов в расчете на 100 строк исходного кода программного средства, $Н_м$ = 3%.

Расходы по статье “Спецоборудование” включают затраты средств на приобретение вспомогательных специального назначения технических и программных средств, необходимых для разработки конкретного программного средства, включая расходы на их проектирование, изготовление, отладку, установку и эксплуатацию. Так как при разработке анализируемого проекта не было запланировано и использовано никакое спецоборудование, то расчеты по данной статье не производятся.

Расходы по статье “Машинное время” включают оплату машинного времени, необходимого для разработки и отладки программного средства, которое определяется по нормативам (в машино-часах) на 100 строк исходного кода машинного времени в зависимости от характера решаемых задач и типа ПК. Расчет расходов на оплату машинного времени происходит по формуле:

$$Р_м = Ц_м * \frac{V_о}{100\%} * Н_{мв} = 150 * \frac{5940}{100\%} * 12 = 106920 \text{ руб.}, \quad (7.12)$$

где $Ц_м$ – цена одного машино-часа, $Ц_м$= 150 руб.

$V_о$ – общий объем ПС (строк исходного кода)

$Н_{мв}$ – норматив расхода машинного времени на отладку 100 строк исходного кода, $Н_{мв}$=12.

Так как в период и для разработки данного программного средства не было производственной необходимости для научных командировок, то расчеты по данной статье затрат не производятся.

Расходы по статье “Прочие затраты” включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу в процентах (20%) к основной заработной плате исполнителей. Расчет прочих затрат на разработку программного средства определяется по формуле:

$$П_з = \frac{З_о * Н_{пз}}{100\%} = \frac{7167999{,}68 * 20\%}{100\%} = 1433599 \text{ руб.}, \qquad (7.13)$$

где $Н_{пз}$ – норматив прочих затрат в целом по организации, $Н_{пз}$ = 20%.

Затраты по статье “Накладные расходы”, связанные с расходами на общехозяйственные нужды, на содержание аппарата управления, опытных производств и прочего, определяются в процентном отношении к основной заработной плате исполнителей. Расчет накладных расходов определяется по формуле:

$$Р_н = \frac{З_о * Н_{рн}}{100\%} = \frac{7167999{,}68 * 100\%}{100\%} = 7167999{,}68 \text{ руб.}, \qquad (7.13)$$

где $Н_{рн}$ – норматив накладных расходов в целом, $Н_{рн}$=100%.

Общая сумма расходов по смете на программное средство рассчитывается по формуле:

$$С_п = З_о + З_д + З_{сз} + Н_е + М \quad + Р_м + П_з + Р_н = 7167999{,}68 +$$
$$+716799{,}9 + 2680831{,}8 + 57344 + 358399{,}98 + 106920 +$$
$$+1433599 + 7167999{,}68 = 19689894 \text{ руб.} \qquad (7.14)$$

Рентабельность и прибыль по создаваемому программному средству определяются исходя из результатов анализа рыночных условий, договоренностей с заказчиком и согласования с ним отпускной цены, включая дополнительный налог на добавленную стоимость и отчисления вместный и республиканский бюджеты. Прибыль от реализации ПС заказчику рассчитывается по формуле:

$$П_о = \frac{С_п * У_{пр}}{100\%} = \frac{19689894 * 30\%}{100\%} = 5906968{,}2 \text{ руб.}, \quad (7.15)$$

где $У_{пр}$ – уровень рентабельности ПС, $У_{пр}$= 30%;

$С_п$ – себестоимость ПС.

Прогнозируемая цена без налогов, включаемых в цену, определяется:

$$Ц_п = С_п + П_о = 19689894 + 5906968{,}2 = 25596862{,}2 \text{ руб.} \quad (7.16)$$

Расчет отчислений и налогов в местный и республиканский бюджеты единим платежом:

$$О_{мр} = \frac{Ц_п * Н_{мр}}{100\% - Н_{мр}} = \frac{25596862{,}2 * 3{,}9\%}{100\% - 3{,}9\%} = 1038790{,}45 \text{ руб.}, \quad (7.17)$$

где $Н_{мр}$– норматив отчислений в местный и республиканский бюджеты,

$Н_{мр}$ = 3,9%

Расчет налога на добавленную стоимость по разрабатываемому программному средству:

$$НДС = \frac{(Ц_п + О_{мр}) * Н_{дс}}{100\%} = \frac{(25596862{,}2 + 1038790{,}45)}{100\%} * 20\% = \quad (7.18)$$

= 5327130,53 руб.,

где $Н_{дс}$ – норматив НДС, $Н_{дс}$ = 20%

Расчет прогнозируемой отпускной цены на разрабатываемое программное средство:

$$Ц_о = Ц_п + О_{мр} + НДС = 25596862{,}2 \text{руб} + 1038790{,}45 + \quad (7.19)$$

$$+2962972{,}4 = 29598625{,}05 \text{ руб.}$$

Расчет затрат на освоение программного средства в расчете на три месяца:

$$Р_о = \frac{С_п * Н_о}{100\%} = \frac{19689894 * 10\%}{100\%} = 1968989{,}4 \text{ руб.}, \quad (7.20)$$

где $Н_о$ – норматив расходов на освоение, $Н_о$ = 10%.

Для упрощения расчетов для составления сметы затрат на сопровождение определяются по установленному нормативу от себестоимости ПС (в расчете на год) и рассчитываются по формуле:

$$Р_с = \frac{С_п * Н_с}{100\%} = \frac{19689894 * 20\%}{100\%} = 3937978{,}8 \text{ руб.}, \quad (7.21)$$

где $Н_с$ – норматив расхода на сопровождение, Нс =20%.

Экономический эффект разработчика. Заказчик оплачивает разработчику всю сумму расходов по проекту, включая прибыль. После уплаты налогов из прибыли в распоряжении заказчика остается чистая прибыль от проекта. В таблице 7.2 представлены все рассчитанные показатели.

Таблица 7.2 - Смета затрат

Наименование статьи	Обозначение	Сумма, руб.
Основная заработная плата	$З_о$	7167999,68
Дополнительная заработная плата	$З_д$	716799,9
Отчисления в ФСЗН (35%)	$З_{зс}$	2680831,8
Налог в фонд оплаты труда (единым платежом 4%)	$Н_е$	57343
Расходы на материалы (3%)	М	358399,98
Расходы на оплату машинного времени	$Р_м$	106920
Расходы на прочие затраты (20%)	$П_з$	1433599
Накладные расходы (100%)	$Р_н$	7167999,68
Себестоимость ПС	$С_п$	19689894
Прибыль от реализации ПС заказчику	$П_о$	5906968,2
Цена без учета налогов	$Ц_п$	25596862,2
Налог в местный и республиканский бюджеты (4,2%)	$О_{мр}$	1038790,45
НДС (20%)	$Н_{дс}$	5327130,53
Прогнозируемая отпускная цена	$Ц_о$	29598625,05

7.4 Расчет показателей экономической эффективности ПС

Экономическая эффективность – соотношение финансовых результатов и затрат по проекту, обеспечивающих ожидаемую норму доходности используемых активов.

Оценка коммерческой эффективности проектов ПС в силу ее особой важности для внутренних и внешних пользователей предполагает:

- определение расчетного периода и расчетных шагов проекта;
- обоснование цены ПС;
- определение денежных потоков с включением всех денежных поступлений по проекту в ходе его осуществления;
- учет изменения стоимости денег во времени;
- оценку затрат и результатов по проекту в соответствии с принципом “без проекта” и с “проектом”;
- оценку инфляции и риска;
- учет налогов, сборов, отчислений и льгот, предусмотренных законодательными нормами, действующими в расчетном периоде.

Ввиду того, что программное средство разрабатывалось для одного объекта, чистую прибыль можно считать в качестве экономического эффекта разработчика от одного проекта.

Так как, в данном случае рассматривается ситуация продажи ПС одному заказчику и не предусматривается массовая поставка ПС на рынок, расчет сметы затрат и цены ПС, поставляемого на рынок, производится не будут.

Денежные средства, получаемые и затрачиваемые в разные моменты времени, имеют разную стоимость. Возможность соизмерения разновременных денежных потоков, достигается путем дисконтирования, т.е. приведения денежных потоков к единому времени. Процесс дисконтирования предполагает определение нормы дисконта и коэффициента дисконтирования.

Применительно к рассматриваемому проекту принято акцентировать внимание на следующих факторах риска:

- Ставка процента. В виду того, что ставки процента в рублях пока не воспринимаются как устойчивые, в качестве безрисковой ставки принята ставка по валютным депозитам в размере 8% (r = 8%). Возможное влияние непредвиденных обстоятельств на величину этой ставки оценено премией за риск в пределах 1% (g1 = 1%).

- Уровень инфляции. Инфляционную премию к безрисковой ставке можно принять в размере 20% (s = 20%).

- Рост спроса. Премия за риск падения спроса установлена в размере 1% (g2 = 1%).

- Стабильность дохода. Премия за риск изменения дохода устанавливается в размере 1% (g3 = 1%).

Нормативная ставка дисконта примем $E_н = 15\%$;

Коэффициенты дисконтирования будут определены следующим образом:

2013г. t=0 α_1=1;

2014г. t=1 $\alpha_2 = (1+E)^{1-2} = \frac{1}{1.15} = 0.87$;

2015г. t=2 $\alpha_3 = (1+E)^{1-3} = \frac{1}{1.15^2} = 0.76$;

2016 г. t=3 $\alpha_4 = (1+E)^{1-4} = \frac{1}{1.15^3} = 0.66$.

Чистый дисконтированный доход рассчитывается по формуле 7.22:

$$ЧДД_t = Р_t\alpha_t - З_t\alpha_t, \quad (7.22)$$

где $Р_t$ - чистый доход в году

$З_t$- затраты в году

α_t- коэффициент дисконтирования.

$$ЧДД_1 = 8906968{,}2 * 1 - 14269088{,}34 = -2782895{,}80 \text{ руб.}$$
$$ЧДД_2 = 14269088{,}34 * 0{,}87 - 0 * 0{,}87 = 12414106{,}85 \text{ руб.}$$
$$ЧДД_3 = 14269088{,}34 * 0{,}76 - 0 * 0{,}76 = 10844507{,}13 \text{руб.}$$
$$ЧДД_4 = 14269088{,}34 * 0{,}66 - 0 * 0{,}66 = 9417598{,}30 \text{ руб.}$$

Интегрированный экономический эффект рассчитывается по формуле:

$$Э_{инт} = \sum_{t=1}^{n} ЧДД_t = \sum_{t=1}^{n} (Р_t\alpha_t - З_t\alpha_t), \quad (7.23)$$

Подставляя результаты уравнения (7.22) в выражение (7.23) получаем:

$$Э_{инт} = -2782895{,}80 + 12414106{,}85 + 10844507{,}13 + 9417598{,}30 =$$
$$= 29893316{,}49 \text{ руб.}$$

Данные расчета экономической эффективности сводим в таблицу 7.3.

Таблица 7.3 – Результаты расчёт экономической эффективности

Показатели	Единица измерения	годы			
		2013 (t_0)	2014 (t_1)	2015 (t_2)	2016 (t_3)
1	2	3	4	5	6
Чистая прибыль	руб	8906968,20	12414106,85	10844507,13	9417598,30
Освоение ПС	руб	1968989,40	-	-	-
Сопровождение ПО	руб	3937978,8	-	-	-
Приобретение ПС	руб	19689894	-	-	-
Всего затрат	руб	25596862,2	-	-	-
ЧДД	руб	−2782895,80	12414106,85	10844507,13	9417598,30
ЧДД с нарастанием	руб	−2782895,80	9631211,05	20475718,18	29893316,48
Дисконтирование		1	0,87	0,76	0,66

Чистый дисконтированный доход имеет максимальное значение во втором году в реализации проекта и составляет 12414106,85 руб. Интегрированный экономический эффект за три года составит 29893316,48 руб. Диаграммы изменения ЧДД и интегрированного экономического эффекта показаны на рисунке 7.1 и 7.2.

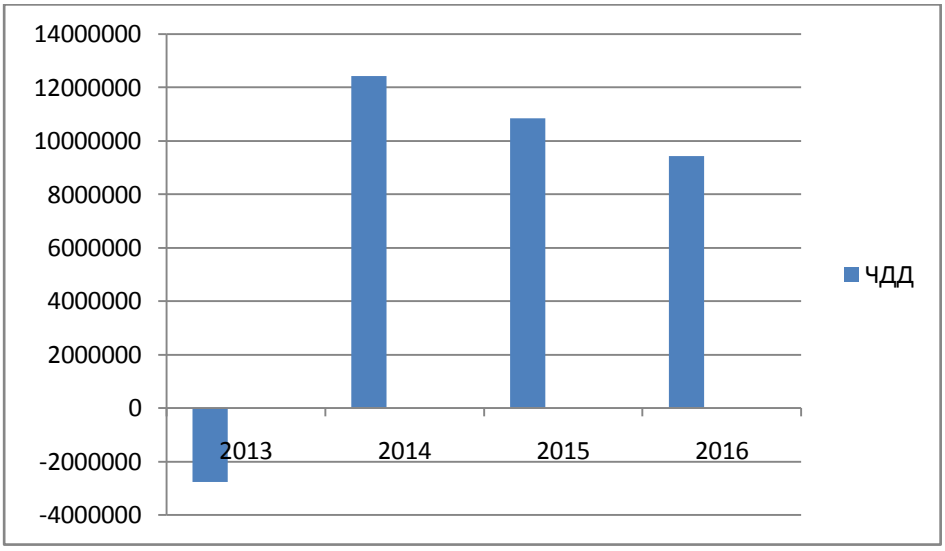

Рисунок 7.1 – Диаграмма чистого дисконтированного дохода

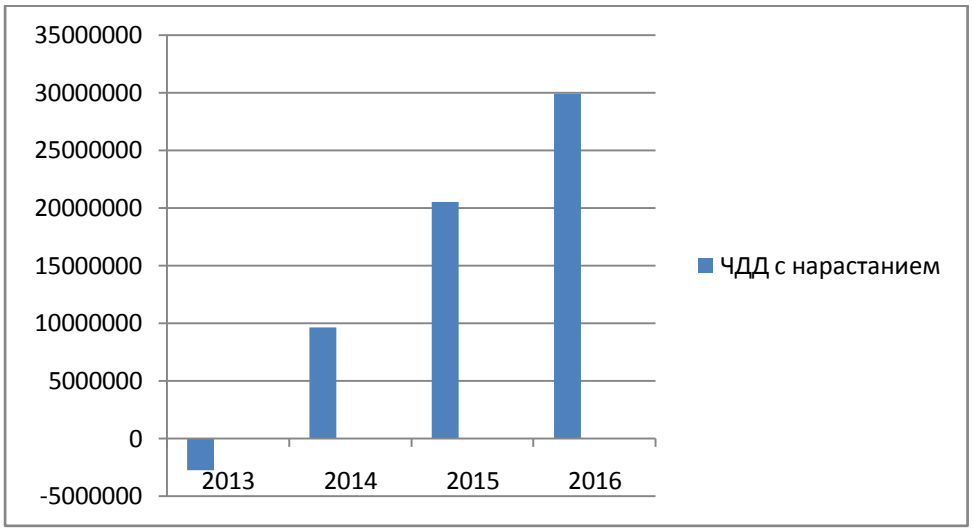

Рисунок 7.2 – Диаграмма интегрированного экономического эффекта

Срок окупаемости, или количество лет, в течении которых инвестиции возвратятся инвестору в виде чистого дохода, рассчитывается по формуле (7.24):

$$T_{ок} = \frac{\sum_{t=1}^{n} З_t * \alpha_t}{P_{ср}} = \frac{8906968{,}20 + 12414106{,}85 + 10844507{,}13 + 9417598{,}30}{1968989{,}40} =$$

$= 4.2$ года. (7.24)

Далее по формуле (7.25) определяем рентабельность инвестиций:

$$P_u = \frac{\sum_{i=1}^{n}(P_t \alpha_t)}{(\sum_{i=1}^{n}(З_t \alpha_t)) * 4} * 100\%, \qquad (7.25)$$

где Pt– чистый доход, полученный в году t, руб.

Зt– затраты (инвестиции) в году t, руб.

αt– коэффициент дисконтирования.

$$P_u = \frac{-2782895{,}80 + 10844507{,}84 + 8276071{,}24 + 5993017{,}1}{25596862{,}2 + 0 + 0 + 0} * 100\% =$$

$= 190\%$.

В процессе технико-экономического обоснования применения разработки программной системы классификации изображений для мониторинга состояния сельскохозяйственной растительности были получены следующие результаты:

- Чистый дисконтированный доход за второй год составил максимальное значение и составил ЧДД =12414106,85 руб.

- Интегрированный экономический эффект за четыре года составил $Э_{инт}$ = = 29893316,49 руб.

- Все дополнительные капитальные затраты на освоение, сопровождение и адаптацию нового ПО окупятся в течение четвёртого года, то есть $Т_{ок}$ = 4,2 года.

- Рентабельность инвестиций составит $Р_{u}$ = 190%.

Таким образом, применение разработки программной системы классификации изображений для мониторинга состояния сельскохозяйственной растительности является эффективной и экономически выгодной для коммерческого успеха.

8 ЭКОЛОГИЧЕСКАЯ БЕЗОПАСНОСТЬ. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ СИСТЕМЫ ОХРАНЫ ОКРУЖАЮЩЕЙ СРЕДЫ СПК «БЕРЕСНИ»

Целью дипломного проекта является разработка программной системы классификации изображений для мониторинга состояния сельскохозяйственной растительности. Система предназначена для обнаружения, выделения и классификации поражений растительности и может использоваться в СПК «Бересни».

Сельскохозяйственный производственный кооператив (СПК) – кооператив, созданный для совместной деятельности по производству, переработке и сбыту сельскохозяйственной продукции. СПК «Бересни» занят в области животноводчества, растениеводства, молочной продукции. Программная система дипломного проекта используется в области растениеводства, чем в свою очередь занят данный СПК. Поэтому имеет смысл применение программной системы классификации изображений для мониторинга состояния сельскохозяйственной растительности на СПК «Бересни». Программная система используется на этапе контроля за состоянием растительности и позволяет оценивать степень поражений, определять здоровые области и поражённые, исходя из чего работники СПК принимают решение о изменении состава удобрений, влияния внешних факторов на растительность, что поможет в дальнейшем избежать появления некачественной продукции и повысить урожайность.

Рассмотрим понятие информационного обеспечения охраны окружающей среды.

С развитием технологий возможны новые способы мониторинга за состоянием окружающей среды. Чем выше уровень технологий, тем быстрее, точнее, качественнее можно отслеживать состояние окружающей среды. Такие проблемы, как загрязнения воздуха, почв, растительности актуальны в наше время как никогда, поэтому требуется мощная, развитая, единая система мониторинга, способная контролировать, принимать решения относительно охраны окружающей среды.

Информационное обеспечение охраны окружающей среды [1] – это сбор, переработка, хранение и обязательно подготовка к использованию информации, которая необходима для оценки состоянии собственно окружающей среды, экологической деятельности и принятия различного рода решений в этой области. Наиболее показательными примерами одного из

направлений информационного обеспечения являются слежение за погодой, сейсмологическими процессами, подготовка метеопрогнозов и другие связанные с этим виды работ. Информационное обеспечение имеет непосредственное отношение к сельскому хозяйству, так как последнее нуждается в качественных средствах контроля, защиты, оценки качества своей деятельности. Факторы, влияющие на сельхозпродукцию, и являются объектами для информационного обеспечения по контролю за состоянием окружающей среды.

Информационное обеспечение охраны окружающей среды включает в себя [2]:

– необходимую информацию как объект обеспечения;

– процессы работы с информацией, завершающиеся ее подготовкой к использованию;

– распределение или предоставление информации возможным пользователям;

– использование информации, влекущее оценку ее качества и полноты.

Вся эта деятельность регламентируется правом. Правовое регулирование информационного обеспечения необходимо, так как оно:

– может задеть различные интересы, включая безопасность страны;

– нуждается в четких процедурах и установлении ответственности за возможную неполноту, искажение, сокрытие информации, что может повлечь принятие неверных решений и огромный ущерб.

Информация о взаимодействии с окружающей средой и ее охране возникает уже в результате деятельности человека и существует либо как общедоступная, либо как латентная, нуждающаяся в выявлении, фиксации и осознании. Существующие здесь трудности состоят в том, чтобы: правильно определить информационную потребность, т. е. установить, какая информация, действительно являющаяся экологической, относящейся к охране окружающей среды, нужна людям и может быть ими использована; создать эффективные структуры и четко наладить процессы, необходимые для собирания, фиксации (закрепления) и обработки этой информации; реально обеспечить возможности доступа к информации для ее надлежащего использования; создать все необходимые условия правильной (для данного уровня знаний) оценки информации; своевременно и эффективно использовать информацию на благо общества и природы.

Экологическое право частично решает эти задачи, создавая правовые основы для информационного обеспечения охраны окружающей среды.

Первоочередные задачи экологической политики Республики Беларусь [3] является обеспечение экологически безопасных условий для проживания людей, рациональное использование и охрана природных ресурсов, выработка правовых и экономических основ охраны окружающей среды в интересах настоящего и будущих поколений. Эти задачи реализуются через систему планирования природоохранной деятельности, совершенствования законодательства и системы государственного управления.

Основным государственным исполнительным органом в области охраны окружающей среды является Министерство природных ресурсов и охраны окружающей среды [3,4]. В его компетенцию входит:

− государственный контроль за деятельностью министерств, ведомств, предприятий, учреждений и организаций в области природопользования, охраны окружающей среды, природоохранного законодательства, соблюдения норм экологической безопасности;

− проведение единой научно-технической политики в области охраны окружающей среды и природопользования;

− организация мониторинга природной среды;

− осуществление государственной экологической экспертизы;

− утверждение совместно с органами санитарного надзора нормативов качества окружающей среды;

− выдача и аннулирование разрешений на выбросы (сбросы) загрязняющих веществ и окружающую среду, использование природных ресурсов, захоронение (складирование) отходов;

− разработка природоохранных программ;

− учет и оценка природных ресурсов;

− обеспечение населения информацией о состоянии окружающей среды и принимаемых мерах по ее оздоровлению, организация пропаганды экологических знаний;

− контроль за выполнением Республикой Беларусь обязательств по международным соглашениям, осуществление международного сотрудничества в области окружающей среды.

Главная внутренняя задача Министерства природных ресурсов и охраны окружающей среды в области информатизации является организация оптимального функционирования информационных систем с максимальным использованием средств и возможностей вычислительной техники.

В соответствии с Законом РБ «Об охране окружающей среды» [3] и постановлений Правительства Республики Беларусь в стране была образована Национальная система мониторинга окружающей среды

(НСМОС), организация которой занимается Минприроды. НСМОС включает в себя совокупность систем наблюдений, оценок и прогноза состояния окружающей среды и природных явлений, биологических отзывов на изменение окружающей среды под воздействием естественных и техногенных факторов. Под эгидой НСМОС проходит организация сбора, обработка и представление информации органам управления и хозяйствования для выполнения общегосударственных задач рационального природопользования.

Экологический мониторинг выделяет следующие задачи [4]:

− наблюдение за состоянием окружающей среды;

− анализ состояния окружающей среды и прогнозирование его изменений;

− обеспечение органов управления систематического и оперативной информацией о состоянии окружающей среды;

− разработка прогнозов и предупреждения об изменениях состояния окружающей среды;

− разработка научно обоснованных рекомендаций по управлению экологической ситуацией

Различают общий, оперативный и фоновый экологический мониторинг окружающей среды

Общий мониторинг - оптимальные по количеству параметров наблюдения на пунктах, объединенных в единую информационно-технологическую сеть, позволяющие на основе оценки и прогнозирования состояния окружающей среды я регулярно разрабатывать управленческие решения на всех уровня.

Оперативный мониторинг - наблюдение специальных показателей в целевой сети пунктов в реальном масштабе времени по отдельным объектам, источниками повышенного экологического риска в отдельных регионах, я которые определены как зоны чрезвычайной экологической ситуации, а также в районах аварий с вредными экологическими последствиями с целью обеспечения оперативного реагирования на кризисные ситуации и принятия решения по их ликвидации, создания безопасных условий для населения.

Фоновый мониторинг - специальные высокоточные наблюдения за всеми составляющими окружающей среды, а также по характеру, составу, кругооборотом и миграцией загрязняющих веществ, за реакцией организма на загрязнение на уровне отдельных популяций, экосистем и биосферы в целом Этот мониторинг осуществляется в природных и биосферных заповедниках, на других охраняемых территориях.

НСМОС обеспечивает [4] информацией хозяйственные субъектов и население страны о состоянии окружающей среды, городских и сельских территорий, выбросах и сбросах в окружающую среду, климатических и погодных изменениях.

В соответствии с постановлением Совета Министров РБ «О локальном мониторинге окружающей среды» в рамках НСМОС больше чем на 100 экологически опасных предприятиях РБ происходит ведение локального мониторинга. Он проверяет выполнение субъектами хозяйствования постоянных наблюдений за сбросами и выбросами, состоянием подземных вод. Сейчас в системе Минприроды существует эффективная аналитическая служба, которая решает задачи в области экологического лабораторного контроля.

Рассмотрим использование Национальной системы мониторинга окружающей среды в деятельности СПК «Берести».

СПК «Берести» работает в области животноводчества, растениеводства, молочной продукции. Развитая инфраструктура требует наличия и использования специальных средств по мониторингу окружающей среды. Это необходимо для комплексной оценки качества продукции. СПК получает информацию от Национальной системы мониторинга о состоянии атмосферы, загрязнённости почвы и действует, исходя из полученных данных, принимает решения по корректировке своей работы. Это может быть как изменение внутренних факторов, таких, как удобрения и полив, так и внешних – освещённости, температуры, влажности. Эти мероприятия помогают сохранить на должном уровне продукцию СПК «Берести», не теряя в объёме производства.

Таким образом, национальная система мониторинга окружающей среды способствует сохранению природы и её ресурсов на современном уровне. Важность показаний данной системы необходима в сельском хозяйстве. Обеспечение экологической безопасности в сельскохозяйственном производстве приобретает особую актуальность в связи с постоянным развитием агропромышленного комплекса и, как следствие, усилением вредного воздействия на окружающую среду. Мониторинг позволяет контролировать и оценивать состояние сельхозпродукции исходя из географического положения, выращиваемых культур и других факторов. Исходя из понятия мониторинга и целей мониторинга окружающей среды, можно сказать, что программное средство дипломного проекта в какой-то мере попадает под понятие мониторинга, так как оно способно выделять и распознавать поражённые участки растительности, что позволяет

предотвращать распространение заболеваний на здоровые культуры и позволяет сохранить качество продукции на должном уровне.

Имея отношение к мониторингу, программная система классификации изображений для мониторинга состояния сельскохозяйственной растительности способна в комплексе точного земледелия работать, как средство оценки общей заболеваемости, на основании чего можно судить об общей картине состояния растительности на данном СПК.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта был разработан и реализован программная система классификации изображений для мониторинга за состоянием сельскохозяйственной растительности. Данная система выполнялась в качестве экспериментальной работы с целью повышения знаний и приобретение опыта разработки программ в области обработки изображений. В рамках данного дипломного проекта были решены следующие задачи:

- рассмотрены основные характеристики систем поиска объектов на изображении;
- проведен обзор и анализ известных решений;
- рассмотрены существующие методы сегментации изображении;
- спроектирована структурная и функциональная модели системы;
- разработан интерфейс программы;
- реализована система классификации изображений;
- произведены экономические расчеты, обоснована рентабельность системы ;
- рассмотрены основные моменты по экологической безопасности;
- подведены итоги по проделанной работе;
- подготовлена пояснительная записка и графический материал по теме дипломного проекта.

Программа является упрощенным аналогом современных систем из области обработки и анализа изображений и может быть успешно внедрена в производство, а так же в сферы научной деятельности. В ходе выполнения испытаний программа показала свою работопригодность для использование как самостоятельное решение.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Якушев М.В., Якушев В.В. Информационное обеспечение точного земледелия / В.П. Якушев. – СПб.: Питер Ком, 2007.

[2] Михайленко И.М. Управление системами точного земледелия / И.М. Михайленко, 2005.

[3] Вудс Г. Цифровая обработка изображений / Г. Вудс. – М.: Техносфера, 2006.

[4] Яне Б. Обработка изображений / Б. Яне. – М.: Техносфера, 2007.

[5] Санкина Е.М. Защита растений. Фитопатология. Учебное пособие для самостоятельного изучения дисциплины / Е.М. Санкина. – Нижний Новгород.: НГСХА, 2005.

[6] Bradsky G., Kaehler A. Learning OpenCV – O'Reilly, 2008.

[7] Пахомов Б. C++ и MS Visual C++ 2010 / Б. Пахомов – Санкт-Питербург.: БХВ-Питербург, 2011.

[8] Дейтл Х. Как программировать на C++, 5-е издание, 2008.

[9] Носенко А.А. Технико-экономическое обоснование дипломных проектов учеб.-метод. пособие: в 4ч., Ч4: Проекты программного обеспечения . / А.А. Носенко, А.В. Грицай - Минск: БГУИР, 2002.

[10] Палицын В.А. Технико-экономическое обоснование дипломных проектов. : учеб.-метод . пособие: в 4ч. / В.А Палицин. – Минск: БГУИР, 2005.

[11] Михнюк, Т. Ф. Охрана труда и основы экологии : учеб. пособие / Т. Ф. Миханюк – Минск : Выш. шк., 2007.

[12] Девисилов, В. А. Охрана труда: учебник. / В. А. Девисилов. 2-е изд. испр. и доп. – М. : Форум, ИНФРА – М., 2006.

[13] Об охране окружающей среды : Закон Респ. Беларусь, 26 нояб. 1992 г., № 1982-XII : в ред. Закона Респ. Беларусь, 17 июля 2002 г. № 126-З : с изм. и доп.

[14] Научно-методические основы организации и ведения национальной системы мониторинга окружающей среды РБ / А. Т. Войтов [и др.]. – Минск, 2000.

ПРИЛОЖЕНИЕ А
(обязательное)
Пример кода выделения объекта

```
// модуль выделения объектов
namespace ObjectsDetection
{
    /// <summary>
    /// класс реализующий влгоритм выделения объектов
    /// </summary>
    public class ObjectsDetecter : ObjectsDetection.IObjectsDetecter
    {


        /// <summary>
        /// величена определяющая сколько необходимо пикселей
        /// для образования объекта
        /// </summary>
        private readonly int _minBlobArea = 400;
        /// <summary>
        /// максимальное количестов объектов которое
        /// способен выделит алгоритм
        /// </summary>
        private readonly int _maxSameBlobDestination = 70;
        /// <summary>
        /// окрестность которая будет строится для заданного пикселя
        /// </summary>
        private readonly int _maxDeltaBlob = 15;

        /// <summary>
        /// Конструктор создаёт объект выделения объектов
        /// </summary>
        /// <param name="minBlobArea">параметр указывающий
        /// минимальное количество
        /// точек необходимых для образования объекта</param>
        /// <param name="maxDeltaBlob">параметр указывает
        /// окрестность, которая
        /// будет строится для заданного пикселя </param>
        public ObjectsDetecter(int minBlobArea, int maxDeltaBlob)
        {
            _minBlobArea = minBlobArea;
            _maxDeltaBlob = maxDeltaBlob;
        }
```

```
/// <summary>
/// Метод извлечения объектов из битовой маски
/// </summary>
/// <param name="image">битавая маска кадра</param>
```

Продолжение приложения А

```
/// <param name="partitionIndex"></param>
/// <returns></returns>
public List<Blob> ExtractBlobs(Bitmap image)
{

   List<Blob> blobs = new List<Blob>();
   int height = image.Height;
   int width = image.Width;
   int xStartIndex = 0;
   int yStartIndex = 0;


   // получаем масив бит из маски кадра
   BitmapData data = image.LockBits(
         new Rectangle(0, 0, width, height),
         ImageLockMode.ReadOnly, PixelFormat.Format8bppIndexed);

   Dictionary<Point, object> foregroundPixels =
                                                   new Dictionary<Point, object>();
   unsafe
   {
      // получаем указатель на начало битовой маски
      byte* ptr = (byte*)data.Scan0.ToPointer();
      // пробегаем каждый пиксель в битовой маске
      for (int i = yStartIndex; i < height; i++)
      {
         for (int j = xStartIndex; j < width; j++)
         {
            // проверяем принадлежит ли данный
                              // пиксель области которая двигалась
            if (*(ptr + j + i * width) == 255)
            {
            // проверяем рассматривали ли мы уже данный пиксель
              if (!foregroundPixels.ContainsKey(new Point(j, i)))
                {
                   Blob newBlob = new Blob(_maxDeltaBlob);
                   blobs.Add(newBlob);
                   // строим окрестность для пикселя
                                ProcessPixel(j, i, width, height, ptr,
                               foregroundPixels, newBlob);
                }
            }
```

```
            }
        }
    }
    // возвращаем список полученных объектов
    return blobs;
```



```
}

/// <summary>
/// Алгоритм вычисления координат объекта.
/// Позволяет выделить объект даже с присутствием шумов.
/// Метод основон на рассмотрении окрестности для текучего пикселя.
/// </summary>
/// <param name="x">Текущая позиция по X</param>
/// <param name="y">Текущая позиция по Y</param>
/// <param name="imageWidth">Ширина изображения</param>
/// <param name="imageHeigth">Высота изображения</param>
    /// <param name="imageDataPtr">Текущий рассматривоемый пиксель</param>
    /// <param name="foregroundPixels">Текущий моссив пикселей для объекта
    </param>
/// <param name="currentBlob">Текущий объект</param>
    private unsafe void ProcessPixel(int x, int y, int imageWidth,
            int imageHeigth, byte* imageDataPtr,
            Dictionary<Point, object> foregroundPixels, Blob currentBlob)
{
    // Создание точки, которая описывает информацию о пикселе.
    Point addedPoint = new Point(x, y);

    // Добовление текущей информации о пикселе к масиву точек объекта
    currentBlob.AddPoint(addedPoint);

    // Добавить данный пиксель в масив рассмотренных пикселей
    foregroundPixels.Add(addedPoint, null);

    // Рассмотрение окрестности текущего пикселя
    if (y >= 1 && *(imageDataPtr + x + (y - 1) * imageWidth) == 255)
    {
        // Проверяем рассатривали ли мы уже данный пиксель
        if (!foregroundPixels.ContainsKey(new Point(x, y - 1)))
        {
                ProcessPixel(x, y - 1, imageWidth, imageHeigth,  imageDataPtr,
                foregroundPixels, currentBlob);
        }
    }
        if (y < imageWidth - 1 && *(imageDataPtr + x + (y + 1) *  imageWidth) == 255)
    {
        // Проверяем рассатривали ли мы уже данный пиксель
        if (!foregroundPixels.ContainsKey(new Point(x, y + 1)))
```

```
    {
                ProcessPixel(x, y + 1, imageWidth, imageHeigth, imageDataPtr, foregroundPixels,
        currentBlob);
    }
}
```

Продолжение приложения А

```
if (x >= 1 && *(imageDataPtr + x - 1 + y * imageWidth) == 255)
{
    // Проверяем рассатривали ли мы уже данный пиксель
    if (!foregroundPixels.ContainsKey(new Point(x - 1, y)))
    {
                ProcessPixel(x - 1, y, imageWidth, imageHeigth, imageDataPtr,
            foregroundPixels, currentBlob);
    }
}
        if (x < imageHeigth - 1 && *(imageDataPtr + x + 1 + y * imageWidth) == 255)
{
    // Проверяем рассатривали ли мы уже данный пиксель
    if (!foregroundPixels.ContainsKey(new Point(x + 1, y)))
    {
            ProcessPixel(x + 1, y, imageWidth, imageHeigth, imageDataPtr,
            foregroundPixels, currentBlob);
    }
}
        if (x >= 1 && y >= 1 && *(imageDataPtr + x - 1 + (y - 1) * imageWidth) == 255)
{
    // Проверяем рассатривали ли мы уже данный пиксель
    if (!foregroundPixels.ContainsKey(new Point(x - 1, y - 1)))
    {
            ProcessPixel(x - 1, y - 1, imageWidth, imageHeigth, imageDataPtr,
            foregroundPixels, currentBlob);
    }
}
        if (x < imageHeigth - 1 && y < imageWidth - 1 && *(imageDataPtr + x + 1 + (y + 1) *
        imageWidth) == 255)
{
    // Проверяем рассатривали ли мы уже данный пиксель
    if (!foregroundPixels.ContainsKey(new Point(x + 1, y + 1)))
    {
            ProcessPixel(x + 1, y + 1, imageWidth, imageHeigth, imageDataPtr,
            foregroundPixels, currentBlob);
    }
}
        if (x >= 1 && y < imageWidth - 1 && *(imageDataPtr + x - 1 + (y + 1) * imageWidth)
        == 255)
{
    // Проверяем рассатривали ли мы уже данный пиксель
    if (!foregroundPixels.ContainsKey(new Point(x - 1, y + 1)))
```

```
            {
                        ProcessPixel(x - 1, y + 1, imageWidth, imageHeigth, imageDataPtr,
                        foregroundPixels, currentBlob);
            }
        }
```

Продолжение приложения А

```
                if (x < imageHeigth - 1 && y >= 1 && *(imageDataPtr + x + 1 + (y - 1) * imageWidth)
                == 255)
        {
            // Проверяем рассатривали ли мы уже данный пиксель
            if (!foregroundPixels.ContainsKey(new Point(x + 1, y - 1)))
            {
                        ProcessPixel(x + 1, y - 1, imageWidth, imageHeigth, imageDataPtr,
                        foregroundPixels, currentBlob);
            }
        }
    }
    /// <summary>
    /// Метод объединяет пересекающиеся объекты в один объект
    /// </summary>
    /// <param name="blobs">Список выделеных объектов</param>
    public void MergeBlobs(List<Blob> blobs)
    {
        List<Blob> deletedBlobs = new List<Blob>();
        // перебераем все объекты
        for (int i = 0; i < blobs.Count; ++i)
            for (int j = i + 1; j < blobs.Count; ++j)
            {
                // проверяем пересекаются ли объекты
        if(blobs[i].WillRectangle.IntersectsWith(blobs[j].WillRectangle))
                {
                    // объединяем новый объект с текущим
                    blobs[j] = blobs[j] + blobs[i];
          // удаляем объект из списка так как мы его объединили с текущим
                    deletedBlobs.Add(blobs[i]);
                    break;
                }
            }
        // производим удаление объектов которые
        // были объединены из списка всех объектов
        foreach (Blob blob in deletedBlobs)
        {
            blobs.Remove(blob);
        }
    }
  }
}
```

ПРИЛОЖЕНИЕ Б
(обязательное)
Пример кода вторичной сегментации

{

```
he = imread('hestain.png');
imshow(he), title('H&E image');
text(size(he,2),size(he,1)+15,...
    'Image courtesy of Alan Partin, Johns Hopkins University', ...
    'FontSize',7,'HorizontalAlignment','right');
cform = makecform('srgb2lab');
lab_he = applycform(he,cform);
ab = double(lab_he(:,:,2:3));
nrows = size(ab,1);
ncols = size(ab,2);
ab = reshape(ab,nrows*ncols,2);

nColors = 3;
% repeat the clustering 3 times to avoid local minima
[cluster_idx cluster_center] = kmeans(ab,nColors,'distance','sqEuclidean', ...
                                'Replicates',3);
pixel_labels = reshape(cluster_idx,nrows,ncols);
imshow(pixel_labels,[]), title('image labeled by cluster index');
segmented_images = cell(1,3);
rgb_label = repmat(pixel_labels,[1 1 3]);

for k = 1:nColors
   color = he;
   color(rgb_label ~= k) = 0;
   segmented_images{k} = color;
end

imshow(segmented_images{1}), title('objects in cluster 1');
imshow(segmented_images{2}), title('objects in cluster 2');
```

```
imshow(segmented_images{3}), title('objects in cluster 3');
mean_cluster_value = mean(cluster_center,2);
[tmp, idx] = sort(mean_cluster_value);
blue_cluster_num = idx(1);
```

Продолжение приложения Б

```
L = lab_he(:,:,1);
blue_idx = find(pixel_labels == blue_cluster_num);
L_blue = L(blue_idx);
is_light_blue = im2bw(L_blue,graythresh(L_blue));
nuclei_labels = repmat(uint8(0),[nrows ncols]);
nuclei_labels(blue_idx(is_light_blue==false)) = 1;
nuclei_labels = repmat(nuclei_labels,[1 1 3]);
blue_nuclei = he;
blue_nuclei(nuclei_labels ~= 1) = 0;
imshow(blue_nuclei), title('blue nuclei');
}
```

ПРИЛОЖЕНИЕ В
(обязательное)
Пример кода пирамидальной сегментации

```
block_size = 1000;
   storage = cvCreateMemStorage(block_size);
   comp = NULL;
   levels = 2;
   thresh1 = 50;
   thresh2 = 50;

   src->width = dst->width = (image->width & -(1 << levels));
   src->height = dst->height = (image->height & -(1 << levels));

   cvPyrSegmentation(src, dst,
         storage, &comp,
         levels, thresh1, thresh2);

   int n_comp = comp->total;

   map<int, int> mapping;
   for (int i = 0; i < n_comp; i++) {

      CvConnectedComp* cc = (CvConnectedComp*)cvGetSeqElem(comp, i);
      mapping.insert(pair<int, int>(cc->value.val[0], i));
   }
```