**СОДЕРЖАНИЕ**

[Введение 7](#_Toc451560283)

[1 Обзор литературы 9](#_Toc451560284)

[1.1 Актуальность точного земледелия 9](#_Toc451560285)

[1.2 Технологии создания актуальных карт 9](#_Toc451560286)

[1.3 Основные признаки заболевания растений 12](#_Toc451560287)

[1.4 Методы обработки изображений. Сегментация 13](#_Toc451560288)

[1.5 Существующие разработки в области корректировки карт заболеваемости 15](#_Toc451560289)

[1.6 Аппаратная база для решения задачи 21](#_Toc451560290)

[2 Разработка структурной схемы 23](#_Toc451560291)

[2.1 Выбор направления работы и обоснование выбора 23](#_Toc451560292)

[2.2 Выбор аппаратной части модуля 24](#_Toc451560293)

[2.3 Программная часть модуля 25](#_Toc451560294)

[3 Функциональное проектирование 27](#_Toc451560295)

[3.1 Реализация основных частей модуля 27](#_Toc451560296)

[3.2 Подсистема выделения участка 27](#_Toc451560297)

[3.3 Подсистема сравнения 38](#_Toc451560298)

[3.3 Подсистема формирования данных для отправки 39](#_Toc451560299)

[3.4 Расчет адаптера бортовой системы питания 42](#_Toc451560300)

[4 Разработка программных модулей 46](#_Toc451560301)

[4.1 Основные модули 46](#_Toc451560302)

[4.3 Реализация модуля определения цвета 46](#_Toc451560303)

[4.4 Реализация модуля выделения объекта 48](#_Toc451560304)

[5 Программа и методика испытаний 50](#_Toc451560305)

[6 Руководство пользователя 52](#_Toc451560306)

[6.1 Требования к аппаратной составляющей 52](#_Toc451560307)

[6.2 Руководство к установке операционной системы на ОК Raspberry Pi 53](#_Toc451560308)

[7 Расчет технико-экономического обоснования экономической эффективности инвестиционной оценки технологических процессов использования разработки и внедрения аппаратно-программного модуля корректировки карты состояния растительности для мобильных устройств регистрации данных 59](#_Toc451560309)

[7.1 Характеристика разрабатываемой системы 59](#_Toc451560310)

[7.2 Расчет себестоимости и отпускной цены единицы продукции 59](#_Toc451560311)

[7.3 Расчет интегрированного экономического эффекта в сфере потребления новой техники 63](#_Toc451560312)

[Заключение 66](#_Toc451560313)

[Список использованных источников 67](#_Toc451560314)

[Приложение А (обязательное) Пример кода выделения объекта 69](#_Toc451560315)

[Приложение Б (обязательное) Пример кода вторичной сегментации 75](#_Toc451560316)

[Приложение В (обязательное) Пример кода пирамидальной сегментации 77](#_Toc451560317)

# **ВВЕДЕНИЕ**

Человек восемьдесят процентов информации об окружающем мире воспринимает при помощи зрения. Незначительную часть из всей воспринимаемой информации человек способен осознать. Еще меньшую часть он способен формализовать, что бы ее 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 концепции лежит справедливое утверждение о том, что поле никогда не бывает абсолютно однородным. Это всегда «лоскутное одеяло», где на соседних участках, площадью несколько десятков квадратных метров, количество органики, минеральных веществ, влаги может существенно отличаться. Из-за особенностей рельефа разной бывает также температура почвы, освещенность и скорость ветра в приземном слое. Все это ставит растения в неравные условия. Но традиционной агротехникой такие тонкости практически не учитываются, а потому одинаковую дозу удобрений, минерального питания, средств защиты растений получают все. В результате из-за несоблюдения оптимальной нормы внесения химических и биологических препаратов, усредненности технологий обработки почвы и ухода за посевами страдает и урожайность, и экология, и сам земледелец несет дополнительные расходы, вернее, лишается возможности экономить.

В основе научной концепции точного земледелия лежат представления о существовании [неоднородностей в пределах одного поля](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D0%BE%D1%87%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BF%D0%BE%D0%BA%D1%80%D0%BE%D0%B2%D0%B0). Для оценки и детектирования этих неоднородностей используются новейшие технологии, такие как системы глобального позиционирования ([GPS](http://ru.wikipedia.org/wiki/GPS), [ГЛОНАСС](http://ru.wikipedia.org/wiki/%D0%93%D0%9B%D0%9E%D0%9D%D0%90%D0%A1%D0%A1)), специальные датчики, аэрофотоснимки и [снимки со спутников](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%81%D0%BC%D0%BE%D1%81%D0%BD%D0%B8%D0%BC%D0%BA%D0%B8), а также специальные программы для агроменеджмента на базе [геоинформационных систем](http://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) (ГИС). Собранные данные используются для планирования высева, расчета 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;

- провести моделирование 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

# **ОБЗОР ЛИТЕРАТУРЫ**

* 1. **Актуальность точного земледелия**

В последнее десятилетие внимание аграрной науки всего мира привлечено к проблеме разработки методологии, так называемого, «точного» земледелия (precision farming). Появление болезней у растений лучше всего предотвращать, нежели потом тратить усилия и средства для борьбы с ними. В качестве профилактических мер следует тщательно соблюдать дозировку удобрений. Недостаток их может привести к снижению иммунитета, слабому росту растений, передозировка вообще может сжечь, погубить растение.

Суть точного земледелия заключается в том, что обработка полей производится в зависимости от реальных потребностей, выращиваемых в данном месте культур. Эти потребности определяются с помощью современных информационных технологий, включая космическую съемку и методы ДЗЗ (дистанционного зондирования земли). При этом средства обработки дифференцируются в пределах различных участков поля, давая максимальный эффект при минимальном ущербе окружающей среде и снижении общего расхода применяемых веществ [1].

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) [2].

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 Технологии создания актуальных карт**

В том, что владение точной и достоверной информацией есть важнейшее условие достижения успеха, уже никого не нужно убеждать. Но еще более важно уметь работать с имеющейся информацией. Методы работы с данными постоянно совершенствуются, и теперь уже привычно видеть документы, таблицы, графики, чертежи и картинки на экране компьютера.

Существуют виды деятельности, в которых карты - электронные, бумажные или хотя бы представляемые в уме - незаменимы. Ведь многие дела невозможно начать, не выяснив предварительно, где именно находится точка приложения наших усилий. В том числе в точном земледелии.

Последние десятилетия широко развивались области применения карт, и связано это с возникновением Географических Информационных Систем, воплотивших принципиально новый подход в работе с пространственными данными.

Географическая Информационная Система (ГИС) - это вычислительная система, позволяющая показывать необходимые данные на электронной карте. На карты ГИС можно нанести не только географические, но и статистические, демографические, технические и многие другие виды данных и применять к ним разнообразные аналитические операции.

Электронная карта, созданная в ГИС, поддерживается мощным арсеналом аналитических средств, богатым инструментарием создания и редактирования объектов, а также базами данных, специализированными устройствами сканирования, печати и другими техническими решениями, средствами Интернет - и даже космическими снимками и информацией со спутников.

С помощью ГИС природоохранные организации следят за состоянием лесов, рек и почв. Коммунальные службы планируют и проводят мероприятия по обслуживанию городских сетей. Спасатели, пожарники и ремонтники оперативно рассчитывают оптимальные маршруты [3].

Дистанционное зондирование и фотограмметрия направлены на получение информации об объектах по изображениям, полученным с летательного аппарата. Существует два вида дистанционного зондирования – это аэросъемка и космическая съемка. В настоящее время имеется большое количество съемочных систем воздушного базирования: Leica ADS40, ADS60, ADS80; Vexcel UltraCamX, UltraCamD, UltraCamL; Rollei Mic Pro 39 MPix, Mic Pro 60 MPix и спутниковых систем GeoEye-1, Ikonos, WorldView, OrbView-3, Alos, Spot-5, Terra, RadarSat и многие другие. Эти системы применяются для решения различных задач: картографирование территорий в масштабах от 1:2 000 до 1:1 000 000, лесоустройство, океанология, экология, ликвидация чрезвычайных ситуаций, угольная промышленность, нефтегазовая отрасль, метеорология и другие.

Методы дистанционного зондирования, кроме получения метрической информации об объекте, позволяют определить и качественные характеристики о нем. Для этих целей используются специальные методы дешифрирования аэрокосмической информации, которые подразделяют на: автоматизированные и интерактивные. Для выполнения более качественного дешифрирования объектов по аэрокосмическим снимкам съемку выполняют в узких спектральных каналах, количество которых может достигать до 250.

В настоящее время 98% аэрокосмической информации обрабатывают на цифровых фотограмметрических станциях, которые представляют собой персональный компьютер со специализированным программным обеспечением и специальным монитором.

Преимуществом дистанционного зондирования перед другими способами получения информации об объектах является производительность (площадь одного снимка может составлять несколько десятков км2), бесконтактность метода и значительное сокращение трудозатрат на полевые работы [4].

В сельскохозяйственной деятельности можно выделить такие виды карт:

1 Карта полей создается на основе снимка высокого разрешения путем векторизации границ полей с последующим ее уточнением по данным систем позиционирования GPS / ГЛОНАСС и является базовой для построения других тематических карт. Карта полей, используя объективные размеры площадей полей, протяженность дорог и других объектов, обеспечивает планирование, учет и контроль всех сельскохозяйственных операций, а также анализ условий, влияющих на рост растений.

2 Агрохимическая карта составляется на основе карты полей и данных (содержание азота, фосфора, калия, гумуса, кислотность, тип почвы, дата проведения анализа и т. д.) агрохимического центра или оперативного анализа почвенных проб пространственно привязанных к элементарным участкам поля. На основе агрохимических данных рассчитываются нормы внесения удобрений и средств защиты на конкретный участок для заданного вида сельскохозяйственной культуры.

3 Карта урожайности строится путем привязки данных о местонахождении комбайна, с информацией о собранном урожае. При этом поле считается состоящим из совокупности элементарных участков, объединенных по урожайности в однородные кластеры (зоны).

На урожайность влияют такие факторы, как плодородие почвы, дозы и виды внесенных удобрений, топография местности, технология посева, ухода за сельскохозяйственной культурой, уборки урожая, качество семян, болезни, вредители сельскохозяйственных растений, погодные условия и т. д. Поэтому проведение ретроспективного анализа карт урожайности позволяет выявить и учесть показатели, негативно влияющие на урожайность.

Карты полей, агрохимическая и урожайности служат информационной основой для формирования технологической карты на следующий год [5].

**1.3 Основные признаки заболевания растений**

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:

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



Рисунок 1.2 – Пораженные грибковой болезнью листья

2 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, пораженное бактериальной болезнью

3 Здоровые растения, изображены на рисунке 1.4. Они характеризуются однородным зеленым цветом без участков поражений и высохших листьев[6].



Рисунок 1.4 – Пример здорового растения

## **1.4 Методы обработки изображений. Сегментация**

В качестве источника информации о методах обработки изображений были использованы книги Г. Вудса [7] и Б. Яне [8]. 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 все 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:

1 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](http://ru.wikipedia.org/wiki/%D0%AD%D0%B2%D1%80%D0%B8%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC);
* поместить каждый пиксель изображения в кластер, центр которого ближе всего к этому пикселю;
* заново вычислить центры кластеров, [усредняя](http://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B5%D0%B4%D0%BD%D0%B5%D0%B5_%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5) все пиксели в кластере;
* повторять шаги 2 и 3 до сходимости.

2 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 ее 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.

3 Выделение краев. Выделение краев — 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 краев используются как основа для другого метода сегментации. Обнаруженные края часто бывают разорванными. Но чтобы выделить объект на изображении, нужны замкнутые границы области.

Можно выделить следующую методику обработки данных для системы мониторинга состояния растительности в задачах точного земледелия, которая должна обеспечивать решение следующих задач [9]:

1. Составление карт неоднородностей состояния сельскохозяйственной

растительности, учитывающих как состояние растительности (наличие отсутствие заболевания), так и количество зеленой массы растений на отдельных участках поля.

1. Составление карт заболеваний растительности на основе анализа их яркостных характеристик.
2. Составление карт, которые могут использоваться в качестве входных данных для систем внесения средств защиты растений.

Таким образом, для принятия решений о состоянии растительности и количестве вносимых средств защиты растений нужно выполнить следующие действия:

1. Выполнить предварительную обработку: фильтрацию, выравнивание баланса белого, геопривязку данных.
2. Вычислить дополнительные признаки для каждого обрабатываемого изображения: текстурные (5 главных компонент, получаемых из 14 характеристик Харалика), фрактальные (фрактальная размерность) и цветовые (диапазоны значений насыщенности и оттенка различных классов объектов, нормализованные редуцированные гистограммы для обучающей выборки нейросетевого классификатора.
3. Выполнить многокритериальную пороговую и совместную сегментацию цветовых, текстурных и фрактальных характеристик для выделения различных областей на исходных изображениях: растительность (пораженная и здоровая), почва, граница растительности и почвы, посторонние предметы.
4. Обучить предложенный нейросетевой классификатор и распознать аэроснимки для получения карты заболеваемости, показывающей участки почвы, здоровой и пораженной заболеванием растительности и используемой в качестве основы для вычисления статистических показателей урожайности.
5. Сформировать карты уровня заболеваемости, которые являются входными данными для подсистемы управления форсунками техники, осуществляющей внесение необходимого количества средств защиты растений для их лечения.

## **1.5 Существующие разработки в области корректировки карт заболеваемости**

Анализ литературы показал, что существуют схожие с темой дипломного проекта разработки, но не являются полным решением поставленной задачи. Они, как правило, являются частью какой-то системы, и не могут работать независимо. Так же в данных разработках часто используются специальные сканеры и сенсоры, для которых приходиться создавать специальное программное обеспечение и требуют специальных знаний для наладки и ремонта [10].

**1.5.1** Мобильная геоинформационная система электронного учета сельскохозяйственных земель «ГЕО-План» предназначена для построения и корректировки электронных карт сельскохозяйственных полей с помощью GPS-технологий, а также определения фактических границ и площадей обработанной части поля по данным GPS-приёмника.

Функциональные возможности:

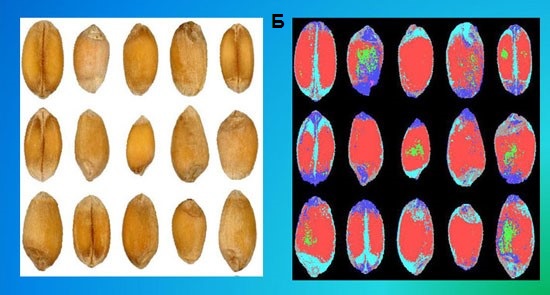
* создание пользовательских схем полей в векторной и растровой форме;
* корректировка текущих схем полей с уточнением их границ, разбиением или объединением;
* ввод GPS-данных с контролем качества по количеству используемых в работе спутников и геометрии их положения, влияющим на точность определения местоположения;
* отображение на схеме в реальном времени получаемых от GPS-данных;
* измерение на схеме расстояний и площадей;
* определение по упрощенной технологии части поля, обработанной сельхозтехникой;
* корректировка сопроводительной информации по каждому полю.

Состав:

* мобильный компьютер;
* высокоточный GPS-приёмник, подключенный к мобильному компьютеру;
* специальное программное обеспечение (ПО).

**1.5.2** В России данные исследования касаются не только заболеваемости полей. Например, для анализа цифрового изображения зерна для оценки его качества применяли программно-аппаратный комплекс (АПК), разработанный специалистами ООО НИЦ «Интеллектуальные сканирующие системы», который состоял из серийно выпускаемого сканера, компьютера и специального программного обеспечения (СПО) Пример промежуточных результатов работы данного АПК представлены на рисунках 1.5 – 1.7 [11].





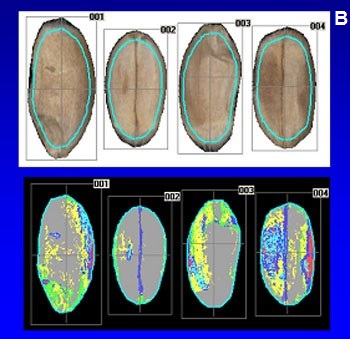


Рисунок 1.5 – Изображения зерновки в процессе разработки методики экспонирования зерна пшеницы: а – зерна «желтобочки» на белом фоне (улучшенный режим); б – формирование зон цветности; в – оконтуривание, выделение центральной части, получение геометрических характеристик

Примером российского аналога можно считать разработки ООО «ЭКО-Разум», которая предоставляют следующие решения:

**1.5.3** Универсальная система картирования Церес предназначена для непрерывного измерения количества собранного урожая и влажности зерна с единицы площади с учетом местоположения комбайна и неровностей поля. Интерфейс системы картирования урожайности Ceres представлен на рисунке 1.6.



Рисунок 1.6 – Интерфейс системы картирования урожайности Ceres

Функциональные возможности системы картирования Церес:

* определяет влажность зерна диэлектрическим датчиком, установленным в тракте движения зерна;
* определяет объемное количество зерна оптическими датчиками, установленным в элеваторе комбайна, а также отображает массу собранного урожая в сухом и влажном состоянии непосредственно в процессе уборки;
* система оснащена встроенным устройством считывания SD-карт, позволяющим отслеживать и наносить на карту урожайность полей в процессе уборки [12];

**1.5.4** Система GreenStar Harvest Doc (рисунок 1.7) позволяет измерять текущие значения количества зерна и его влажности при уборке урожая на комбайнах John Deere.



Рисунок 1.7 – функциональные модули системы GreenStar Harvest Doc

Система Хавест Док состоит из:

* терминал для комбайнов GreenStar Harvest Doc;
* курсоуказатель StarFire iTC, работающий с поправкой SF1 (точность - 30 см);
* экран ГринСтар 2600;
* ключ активации;
* набор проводов и фиттингов;
* приборы для измерения массы и влажности зерна;
* настольное программное обеспечение FarmWorks (ФармВоркс) для построения карт урожайности и другой обработки данных [13]

ГИС Панорама АГРО предназначена для комплексной автоматизации управления сельскохозяйственным предприятием в отрасли растениеводства и обеспечивает решение двух взаимосвязанных задач: управление аграрными технологиями и мониторинг подвижных технических средств компании на основе GPS/ГЛОНАСС навигации.

Американская компания Trimble предлагает такие решения для аэросъемки[14]:

**1.5.5** Trimble AX80 - высоко производительное, универсальное и полностью интегрированное решение от Trimble для воздушного лазерного сканирования, наиболее соответствующее потребностям отрасли (рисунок 1.8). С помощью мощной лазерной системы, характеризующейся частотой повторения лазерных импульсов (PRR) 800 кГц, сканер захватывает плотные облака точек с высоком разрешением. Решение также использует такие передовые методы как одновременная обработка большого количества импульсов, оцифровка эхо-сигналов и анализ формы волны. Оснащенный программным обеспечением для планирования полета (Trimble flight planning), программой управления сенсором и программой Trimble Inpho для обработки данных, AX80 представляет собой законченное решение, отличающееся непревзойденной производительностью, эксплуатационной гибкостью, эффективностью, и надежностью.

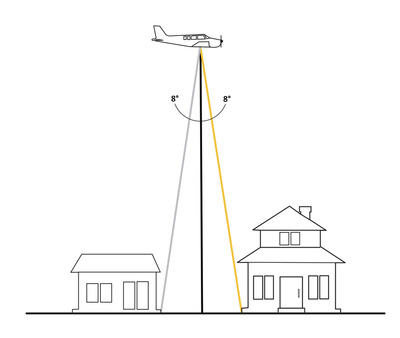
 

Рисунок 1.8 – Внешний вид и схема действия Trimble AX80

**1.5.6** DSS-500 высокопроизводительное законченное решение, разработанное для быстрой и эффективной аэросъемки (рисунок 1.9). DSS-500 производит высокоточные цифровые ортофотопланы и инфракрасные фотоснимки с высоким цветовым разрешением. Все это прибор делает, находясь в воздухе и сразу проверяя качество съемки. В результате, нет необходимости в повторных полетах, что позволяет сэкономить время и деньги.

Рисунок 1.9 – Внешний вид и результат действия Trimble DSS-500

Основные функции:

* расчет и запись координат и высоты сенсора для прямого
* геопозиционирования изображений (и данных ВЛС)
* управление камерой и контроль полета (FMS)
* управление ВЛС (ВКЛ/ВЫКЛ запись точек)
* рассчет в реальном времени элементов внешнего ориентирования для ортофототрансформирования изображений
  1. **Аппаратная база для решения задачи**

По причине особенностей ДЗЗ со спутника [15] целесообразнее создавать модуль для БПЛА. Такой модуль должен обладать малым весом и габаритами, совместимостью с бортовым оборудованием и, желательно, возможностью потенциального расширения или варьирования функций.

Первыми рассмотрим контроллеры. Задачи для контроллеры могут возникать в различных областях — при автоматизации технологических процессов (АСУ ТП), при управлении транспортными системами, для поддержания климата в зданиях. Контроллеры измеряют и регулируют различные физические параметры и логические состояния: температуру, влажность, давление газа или жидкости, значения тока, напряжения, логические состояния системы и так далее [16].

В целом контроллеры созданы для получения небольших массивов информации от датчиков и принятия решения, которое зависит от заложенной в ней программы и отправки управляющих сигналов на агрегаты.

Далее рассмотрим ПЛИС. Программи́руемая логи́ческая интегра́льная схе́ма (ПЛИС, англ. programmable logic device, PLD) — электронный компонент, используемый для создания цифровых интегральных схем. В отличие от обычных цифровых микросхем, логика работы ПЛИС не определяется при изготовлении, а задается посредством программирования (проектирования). Для программирования используются программатор и IDE (отладочная среда), позволяющие задать желаемую структуру цифрового устройства в виде принципиальной электрической схемы или программы на специальных языках описания аппаратуры: Verilog, VHDL, AHDL и другие [17] .

Так же существуют одноплатные компьютеры, которые не уступают по вычислительной мощности настольным и при этом имеют малые габариты. Они имеют уже готовые модули, с предусмотренными для них интерфейсами. Их характеристики [18] позволяют ставить полноценные операционные системы, что позволяет использовать языки программирования высокого уровня. Это упрощает написание программного обеспечения, его отладку и использование конечным пользователем.

В рамках дипломного проекта была рассмотрена линейка от Raspberry Pi. Raspberry Pi выпускается в нескольких комплектациях: модель «A», модель «B», модель «B+» и модель «2 B». Первые три версии оснащены ARM11 процессором Broadcom BCM2835 с тактовой частотой 700 МГц и модулем оперативной памяти на 256МБ/512МБ, размещенными по технологии «package-on-packageruen» непосредственно на процессоре. Модель «2 B» оснащается процессором с 4 ядрами Cortex-A7 с частотой 1ГГц и оперативной памятью размером 1ГБ. Модель «A» оснащается одним USB 2.0 портом, модель «B» двумя, а модели «B+» и «2 B» — четырьмя. Также в моделях «B», «B+» и «2 B» присутствует порт Ethernet. Помимо основного ядра, BCM2835 включает в себя графическое ядро с поддержкой OpenGL ES 2.0, аппаратного ускорения и FullHD-видео и DSP-ядро. Одной из особенностей является отсутствие часов реального времени.

Вывод видеосигнала возможен через композитный разъём RCA или через цифровой HDMI-интерфейс. В версии «B+» и «2B» вывод возможен через аудиоразьем 3,5. Корневая файловая система, образ ядра и пользовательские файлы размещаются на карте памяти SD, MMC, microSD (только в модели «B+») или SDIOruen.

Одной из самых интересных особенностей Raspberry Pi является наличие портов GPIO (general purpose input/output). Благодаря этому компьютер можно использовать для управления различными устройствами. В модели «B» платы присутствуют 26 портов, а в модели «B+» и «2 B» - 40 портов GPIO

1. **РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ**

**2.1 Выбор направления работы и обоснование выбора**

Для разработки аппаратно-программного комплекса необходимо определить требования к самому комплексу:

* аппаратная часть должна быть совместимая с поставленным в задаче оборудованием
* интерфейсы связи и питания бортового оборудования и встраиваемого модуля должны взаимодействовать
* корректировка карт заболеваемости
* формирование сигнала для дальнейшей обработки

Входной информацией для аппаратно-программного модуля будет изображение, полученное со средств регистрации данных устройства, на которое будет установлен модуль.

Выходной информацией будет сигнал, который формируется для дальнейшего анализа состояний растительности. Так же в память будет записываться актуальная информация по состоянию растительности.

Данные должны записываться на сервер и отправляться на систему принятия решений (СПР).

Так как разрабатываемый модуль является частью общей системы нужно показать общую схему работы, принятую в точном земледелии.

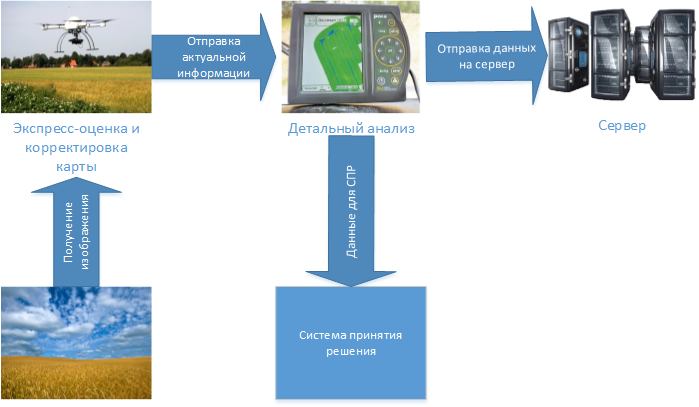


Рисунок 2.1 – Структурная схема общего процесса

**2.2 Выбор аппаратной части модуля**

Конечная цель исследования для беспилотных летательных аппаратов (БЛА) является автономная эксплуатация. Исследования, проведенные в последнее десятилетие высветил потенциал видения зондирования в этой связи. Несмотря на это, жизненно важное значение для выполнения задач, возложенных на любой тип беспилотных летательных аппаратов, дистанционное зондирование земли является более важным для малых летательных аппаратов из-за отсутствия высокоточных инерциальных датчиков. Кроме того, неопределенность сигнала GPS требует более полагаться на техническое зрение для малых транспортных средств. Оффлайн обработка не самый продуктивный вариант с точки зрения автономии, т.к. это предполагает более сложные платформы для осуществления обработки видения на борту не подходящие из-за строгой грузоподъемности и запаса мощности летательных устройств регистрации данных. Существует потребность в новых архитектурах обработки изображений для малых беспилотных летательных аппаратов.

Недавние исследования показали обнадеживающие результаты с ПЛИС на основе жесткой логики. Но из-за цены разработки таких ПЛИС резко возрастает цена конечного изделия. [19] Поэтому предлагается вариант решения на ОК (одноплатном компьютере) для удешевления и ускорения работы с данными на борту устройства регистрации данных. Он не требует дополнительных изменений в конструкции БЛПА, может использовать свои интерфейсы для связи с бортовым оборудование, имеет собственный модуль камеры и требует лишь адаптацию питания.

Из многообразия одноплатных компьютеров решено выбрать Raspberry Pi 2B. Данная модель имеет сравнительно не большую стоимость, обширную документацию и свободно распространяется. Очевидными преимуществами будут дополнительные порты и более лучший процессор по сравнению с предыдущими моделями. Кроме привычных для компьютера интерфейсов Raspberry Pi 2B имеет GPIO (*general-purpose input/output)* — интерфейс для связи между компонентами компьютерной системы, к примеру микропроцессором и различными периферийными устройствами. Контакты GPIO могут выступать как в роли входа, так и в роли выхода — это, как правило, конфигурируется. GPIO контакты часто группируются в порты.

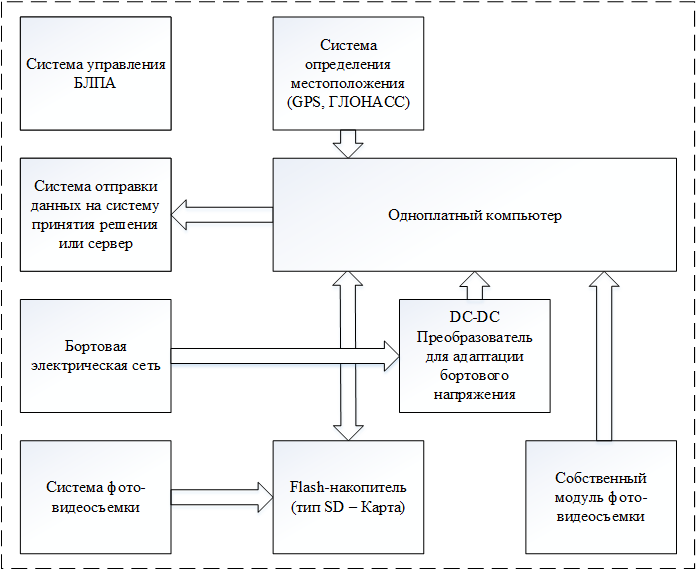


Рисунок 2.2 - Аппаратное представление связей модуля и систем БЛПА

**2.3 Программная часть модуля**

Программная часть выполняет непосредственную часть корректировки карты состояния растительности. Для растений основным признаком состояния здоровья или наличия какой-либо болезни является цветовая характеристика. Следовательно, программная часть должна:

* находить участок для исследования;
* распознавать цвет;
* формировать сигнал для дальнейшей обработки или принятия решения.

За нахождение участка для исследования будет ответственна подсистема выделения исследуемого участка. Для распознавания цвета будет создана подсистема сравнения цветовых характеристик выделенного участка изображения, а для подготовки данных для отправки результата работы будет создана подсистема формирования данных.

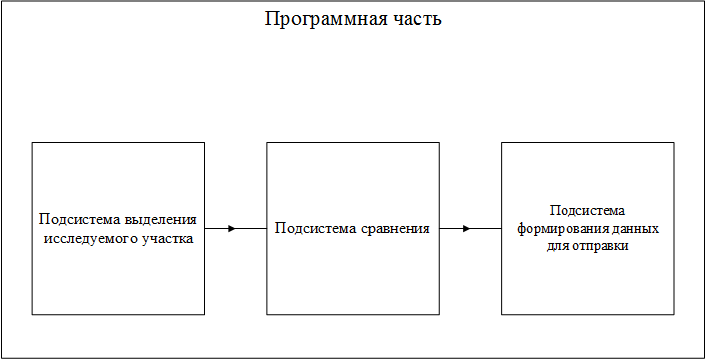


Рисунок 2.3 - Структурная схема программной части модуля

Подсистема выделения участка должна реализовывать поиск и копирование участка для исследования. Она должна определять границу объекта по заданным параметрам и отделить его от остального изображения для дальнейшего анализа. При этом результат работы модуля сохраняется в промежуточном файле для удобства дальнейшей работы самой программы.

Подсистема сравнения осуществляет обработку изображения. Она ведет подсчет пикселей того или иного цвета, определяет здоровые участки и процентное соотношение пикселей здорового сегмента по отношению к общему количеству пикселей. Это процентное соотношение берется как качественный коэффициент.

Подсистема формирования данных зависит от подсистемы сравнения. Она готовит к отправке и отправляет только те данные, которые требуют дополнительных исследований.

# **3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

## **3.1 Реализация основных частей модуля**

На данном этапе были реализованы основные алгоритмы программной части модуля корректировки карт состояния растительности. 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 и аппаратные дополнения, необходимые для работы модуля.

## **3.2 Подсистема выделения участка**

Подсистема выделения участка состоит из блока обработки изображения и блока поиска участка для изучения (рисунок 3.1).

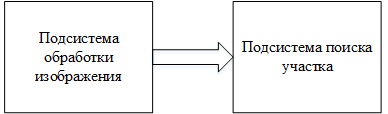


Рисунок 3.1 – Блоки подсистемы выделения участка

Под обработкой изображения подразумевается выравнивание значения баланса белого. Это позволит использовать пространство цветов RGB ([аббревиатура](http://ru.wikipedia.org/wiki/%D0%90%D0%B1%D0%B1%D1%80%D0%B5%D0%B2%D0%B8%D0%B0%D1%82%D1%83%D1%80%D0%B0)  английских слов  Red,  Green,  Blue —[красный](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B0%D1%81%D0%BD%D1%8B%D0%B9_%D1%86%D0%B2%D0%B5%D1%82), [зеленый](http://ru.wikipedia.org/wiki/%D0%97%D0%B5%D0%BB%D1%91%D0%BD%D1%8B%D0%B9_%D1%86%D0%B2%D0%B5%D1%82), [синий](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D0%B8%D0%B9_%D1%86%D0%B2%D0%B5%D1%82)) — [аддитивная](http://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B4%D0%B8%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D1%81%D0%BC%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%D1%86%D0%B2%D0%B5%D1%82%D0%BE%D0%B2) [цветовая модель](http://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C), как правило, описывающая способ синтеза [цвета](http://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82) для [цветовоспроизведения](http://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82%D0%BE%D0%B2%D0%BE%D1%81%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5). [Аддитивной](http://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B4%D0%B8%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D1%81%D0%BC%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%D1%86%D0%B2%D0%B5%D1%82%D0%BE%D0%B2) она называется потому, что цвета получаются путем добавления к черному.

**3.2.1** Изображения, сформированные различными информационными системами, искажаются действием помех. Ослабление действия помех достигается фильтрацией. При фильтрации яркость (сигнал) каждой точки исходного изображения, искаженного помехой, заменяется некоторым другим значением яркости, которое признается в наименьшей степени искаженным помехой. Задача заключается в том, чтобы найти эффективную вычислительную процедуру, которая позволяла бы достигать наилучших результатов. Существует ряд методов и алгоритмов фильтрации, что связано с большим разнообразием сюжетов, которые приходится описывать различными математическими моделями. Кроме того, применяются различные критерии оптимальности, что также ведет к разнообразию методов фильтрации. Наконец, даже при совпадении моделей и критериев очень часто из - за математических трудностей не удается найти оптимальную процедуру.

Простейшие методы обработки изображения – морфологические операции, применяемые к бинарным изображениям. Несмотря на кажущуюся простоту, могут быть полезны для выделения границ и сегментов.

С точки зрения компьютера, изображение содержит сигнал, в котором наряду с полезной информацией содержится шум, от которого необходимо избавиться. Полезную информацию, при этом, желательно так или иначе усилить.

Для решения проблемы когда получается темное или пересвеченное изображение применяют повышение контраста (всего изображения целиком или его части) с помощью линейной коррекцией (или робастной линейной коррекцией). В случае неравномерного освещения, как правило достаточно применения метода single scale retinex (SSR). Однако лучших результатов помогают добиться методы нелинейной коррекции – гамма коррекция или логарифмическая, Multi-Scale Retinex [23].

**3.2.2** Преобразование Хафа позволяет быстро и устойчиво обнаруживать графические примитивы, описываемые аналитическими уравнениями: прямые линии, окружности, эллипсы и т. п. Позднее метод голосования контурных точек в пространство параметров был обобщен и на случай кривых, не описываемых в аналитической форме. В такой наиболее общей форме он получил наименование *обобщенного преобразования Хафа* (GHT).

Рассмотрим сначала задачу обнаружения объекта произвольной формы, заданного эталонным изображением, в случае, когда требуется обеспечить инвариантность результатов обнаружения к сдвигу изображения, но не к его масштабу (рисунок 3.2).

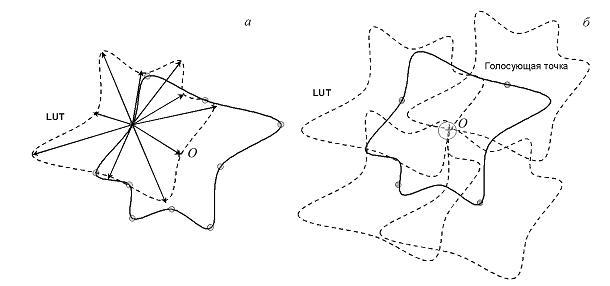
[](http://wiki.technicalvision.ru/index.php/%D0%A4%D0%B0%D0%B9%D0%BB:5-2-7.jpg)

Рисунок 3.2 – Идея обнаружения фигуры произвольной формы, инвариантной к сдвигу: а - обучение путем составления LUT; б - обнаружение путем голосования с использованием LUT

В этом случае, в отличие от задач обнаружения окружности, существенно то, что расстояние *R* от текущего пиксела границы до ее центра больше не константа, а является функцией *R*(*ϕ*) от угла *ϕ* радиуса-вектора, направленного от точки контура к центру (рисунок 3.2 *а*). В дополнение, в общем случае, "центр" здесь должен заново интерпретироваться как некая условная точка локализации *O*. Выбор точки локализации *O* не является единственным и может регулировать ошибки. В общем случае следует ожидать, что положение точки локализации рядом с центром тяжести периметра объекта минимизирует ошибки, обусловленные неточностью оценки ориентации края.

Для определения простых форм функция *R*(*ϕ*) может быть описана аналитически. Однако для большинства форм это невозможно. Тем не менее, Ballard показал, что подход еще остается жизнеспособным, так как для запоминания информации о форме можно использовать специальные *просмотровые таблицы* (look-up-table), содержащие дискретные значения *R*(*θ*) для различных значений углов. Соответственно, алгоритм состоит из этапов обучения детектора Хафа путем составления LUT по эталонному изображению (рисунок 3.2 *а*) и этапа обнаружения объекта на тестовом изображении путем голосования контурных точек с использованием этой LUT (рисунок 3.2 *б*).

Теперь надо обобщить эту схему для случая обнаружения объекта произвольной формы в условиях, когда объект может не только перемещаться, но и вращаться в плоскости изображения. Здесь отталкиваемся от идеи использования информации об ориентации вектора-градиента в контурных точках. В этом случае существенно то, что радиуса-вектор в краевой точке является теперь не функцией от абсолютного угла направления на центр *ϕ*, а функцией относительного угла между направлением градиента и направлением радиуса-вектора (рисунок 3.3). В остальном алгоритм полностью совпадает с описанным выше, однако теперь он уже обладает свойством инвариантности к вращению. Именно в таком, наиболее эффективном виде он и получил название *обобщенного преобразования Хафа* (GHT). Можно также отметить, что за счет использования информации о направлении градиента в точках контура, GHT обладает гораздо лучшей помехозащищенностью, так как точки с неподходящими направлениями градиентов просто не голосуют в пользу соответствующих гипотез, а, следовательно, соотношение голосов, отданных за правильных кандидатов, существенно улучшается.

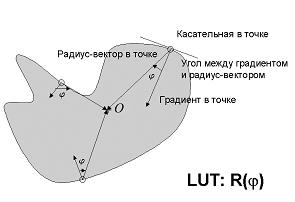
[](http://wiki.technicalvision.ru/index.php/%D0%A4%D0%B0%D0%B9%D0%BB:5-2-8.jpg)

Рисунок 3.3 – Идея обобщенного преобразования Хафа

Таким образом, преобразование Хафа, GHT и их различные модификации обеспечивают инвариантное обнаружение геометрических примитивов и объектов на изображении с высокой степенью помехозащищенности и значительной точностью определения параметров местоположения и ориентации. Существенным ограничением применимости этой группы методов является то, что описанные алгоритмы обнаруживают не сами полутоновые объекты, а их контуры. Поэтому объекты, не имеющие четко выраженного контура, не могут быть подвергнуты детектированию с использованием GHT.

**3.2.3** 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.

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 3.4. Цепные коды применяются для представления границы в виде последовательности отрезков прямых линий определенной длины и направления. В основе этого представления лежит 4- или 8- связная решетка. Длина каждого отрезка определяется разрешением решетки, а направления задаются выбранным кодом. Для представления всех направлений в 4-связной решетке достаточно 2-х бит, а для 8-связной решетки 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:

/\*

Description of the function,

which is used to search for the contours

\*/

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)));

double norm(InputArray src1, int normType=NORM\_L2, InputArray mask=noArray())

//sorting colors

sort( colors.begin(), colors.end(), colors\_sort );

double norm(InputArray src1, InputArray src2, int normType, InputArray mask=noArray())

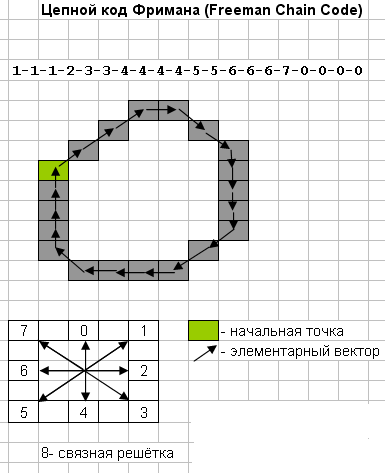


Рисунок 3.4 – Принцип работы цепного кода Фримена

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.

**3.2.4** В качестве основного алгоритма выделения объекта часто 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, помещенная на любой пиксель внутри общей линии водораздела, течет 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 рисунках 3.5 и 3.6 показаны исходное изображение и результат работы алгоритма водораздела.



Рисунок 3.5 – Исходное изображение

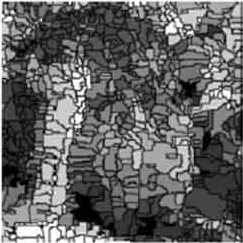


Рисунок 3.6 – Результат работы алгоритма водораздела

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.

**3.2.5** Сегментация является одним из наиболее важных этапов обработки и распознавания изо­бражений, цель которой состоит в выделении на изображениях связных областей (объек­тов) с примерно одинаковыми характеристиками яркости или цветности. Причем в по­следнее время в связи с необходимости автоматической обработки и распознавания все возрастающего потока как статических так и динамических видео данных, наиболее важ­ное значение придается синтезу эффективных процедур сегментации, работающих в ре­жиме реального времени и позволяющих получать надежные помехоустойчивые резуль­таты.

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.

Принцип пирамидальной сегментации показан на рисунке 3.7.

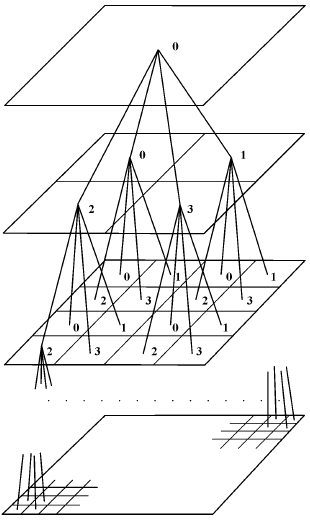


Рисунок 3.7 – Принцип пирамидальной сегментации

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 (3.1):

|  |  |
| --- | --- |
|  | (3.1) |

Здесь (*u; v*) и(*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.

Для корректировки карт состояния растительности необходимо что бы была уже сформированная карта. Для удобства изменения этой карты нужно иметь полное изображение поля, разбитое на отдельные участки, которые будут использоваться для поиска на полученном снимке. Поиск может осуществляться по заранее заготовленным меткам, либо по паттернам, которые характерны данному участку и найдены ранее. Так же можно осуществлять поиск участка по координатам, полученных со спутника.

Поиск по заранее заготовленным меткам является наиболее точным и универсальным, но требует дополнительных действий со стороны пользователя данной системы. Требование к меткам только одно – это четкая видимость на фоне окружения.

Поиск по паттернам имеет меньшую точность, чем поиск по меткам. Но для создания паттернов можно использовать нейронную сеть, которая будет работать автономно. Использование нейронной сети требует большого количества пролетов над полем, поэтому лучше использовать ручные способы задания паттернов. Это требует только некоторого времени работы оператора.

Поиск участка по координатам, полученных со спутника требует постоянной связи со спутниками. Эта связь может ухудшаться из-за погодных условий, а также расходует энергию батарей БЛПА. Так же требуется учитывать время получения координат и время запроса, что приводит к дополнительным вычислениям. Этот способ не требует дополнительных действий со стороны конечного пользователя.

## **3.3 Подсистема сравнения**

Подсистема сравнения включает в себя блок вычисления коэффициента цветности и блок непосредственного сравнения значений, необходимых для экспресс-оценки (рисунок 3.8).

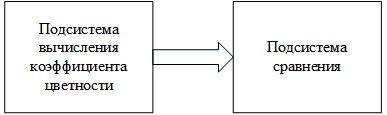


Рисунок 3.8 – Блоки подсистемы сравнения

Предварительная оценка состояния растительности может быть произведена на основе обработки цветовой информации, при этом для снижения влияния условий освещения следует использовать цветовое пространство HSV вместо RGB. Поэтому коэффициенты лучше считать на основе пространства HSV.

Значение оттенка *Hue* вычисляется по формуле:

|  |  |
| --- | --- |
|  | (3.2) |

а насыщенности *Sat* – по формуле:

|  |  |
| --- | --- |
|  | (3.3) |

Оттенок *Hue* может принимать значения от −π/2 до π/2, а насыщенность *Sat* – от 0 до 255. В таблице 3.1 приведенные значения *Hue* и *Sat* для различных типов сегментов (данные о значениях оттенка и насыщенности получены экспертом на основании анализа цветовых характеристик изображений отдельных растений и аэроснимков поля картофеля [20, 21]).

Таблица 3.1 – Значения диапазонов цвета

|  |  |  |
| --- | --- | --- |
| Тип сегмента | Диапазон оттенка Hue | Диапазон насыщенности Sat |
| Здоровый (зеленый) | [0.1; π/2] | [110; 255] |
| Больной (желтый) | [1.0; π/2] | [170; 255] |
| Больной (буро-зеленый) | [1.0; π/2] | [128; 175] |

Данные вычисления проводятся для каждого пикселя. После определения значений производится подсчет количества пикселей здорового сегмента. Так же считается количество пикселей, которые не попали ни в один из диапазонов. Из общего количества пикселей вычитается количество пикселей, которые не попали ни в один из диапазонов. Для вычисления качественного коэффициента берется отношение количества пикселей из здорового сегмента к полученному после вычитания количеству пикселей.

Блок сравнения отвечает за не посредственную экспресс-оценку состояния растительности. Так же здесь фильтруются данные, которые необходимы для дальнейшего анализа. Эталонные значения сформированные на этапе подготовки хранятся в карте памяти и берутся для каждого участка отдельно. В случаях однородного по цвету поля возможно использование одного коэффициента для сравнения.

## **3.3 Подсистема формирования данных для отправки**

Отправка данных нужна для систем поддержки принятия решения. Так же модуль рассматривается в системе точного земледелия, поэтому эти данные можно использовать как часть управляющих сигналов для комплексов более точной оценки. Либо отправлять эти данные непосредственно фермеру для личной детальной оценки состояния растительности. В данной подсистеме имеется блок сохранения данных и блок отправки данных (рисунок 3.9).

Для сохранения и отправки данных будут использованы потоки. Поток считается универсальным средством для передачи данных и может использовать разные архитектуры процессоров. Для передачи данных через интерфейс необходимо будет только трансформировать поток под тот или иной протокол передачи данных.

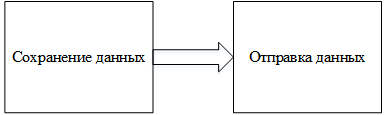


Рисунок 3.9 – Блоки подсистемы формирования данных для отправки

Блок сохранения данных отвечает за сохранение данных в локальное хранилище, для возможности наземной оценки изменений и исключения потери данных при неудачной передаче. Данные для сохранения берутся с блока сравнения. Сохраняются все полученные результаты.

Абстрактный базовый класс *Stream* поддерживает чтение и запись байтов. Класс *Stream* поддерживает асинхронный ввод и вывод. По умолчанию его реализация определяет операции синхронного чтения и записи на основе соответствующих им асинхронных методов, и наоборот.

Все классы, которые работают с потоками, являются производными от класса *Stream*. Класс *Stream* и его производные классы предоставляют способ просмотра источников данных и хранилищ объектов, изолируя программиста конкретных от специфических деталей операционной системы и базовых устройств.

При работе с потоками используются следующие основные операции:

* потоки могут быть считаны. Чтение — это перенос информации из потока в структуру данных, такую как массив байтов.
* в потоки можно вносить записи. Запись — это перенос информации из источника данных в поток.
* потоки поддерживают поиск. Поиск — это выяснение и изменение текущей позиции внутри потока.

В зависимости от лежащих в основе источника или хранилища данных потоки могут поддерживать только некоторые из этих возможностей. Например, [*NetworkStreams*](https://msdn.microsoft.com/ru-ru/library/system.net.sockets.networkstream%28v=vs.100%29.aspx) не поддерживает поиск. Свойства [*CanRead*](https://msdn.microsoft.com/ru-ru/library/system.io.stream.canread%28v=vs.100%29.aspx)*,* [*CanWrite*](https://msdn.microsoft.com/ru-ru/library/system.io.stream.canwrite%28v=vs.100%29.aspx) и [*CanSeek*](https://msdn.microsoft.com/ru-ru/library/system.io.stream.canseek%28v=vs.100%29.aspx) из *Stream* и его производных классов определяют операции, поддерживаемые различными потоками:

1. [*Directory*](https://msdn.microsoft.com/ru-ru/library/system.io.directory%28v=vs.100%29.aspx) предоставляет статические методы операций создания, перемещения и перечисления в директориях и поддиректориях. Класс *DirectoryInfo* предоставляет методы экземпляра.
2. [*DirectoryInfo*](https://msdn.microsoft.com/ru-ru/library/system.io.directoryinfo%28v=vs.100%29.aspx) предоставляет методы экземпляра операций создания, перемещения и перечисления в директориях и поддиректориях. Класс Directory предоставляет статические методы.
3. [*DriveInfo*](https://msdn.microsoft.com/ru-ru/library/system.io.driveinfo%28v=vs.100%29.aspx) предоставляет методы экземпляра для доступа к сведениям о диске.
4. [*File*](https://msdn.microsoft.com/ru-ru/library/system.io.file%28v=vs.100%29.aspx) предоставляет статические методы для создания, копирования, удаления, перемещения и открытия файлов, а также помогает при создании объектов [*FileStream*](https://msdn.microsoft.com/ru-ru/library/system.io.filestream%28v=vs.100%29.aspx). Класс *FileInfo* предоставляет методы экземпляра.
5. [*FileInfo*](https://msdn.microsoft.com/ru-ru/library/system.io.fileinfo%28v=vs.100%29.aspx) предоставляет методы экземпляра для создания, копирования, удаления, перемещения и открытия файлов, а также помогает при создании объектов [*FileStream*](https://msdn.microsoft.com/ru-ru/library/system.io.filestream%28v=vs.100%29.aspx). Класс *File* предоставляет статические методы.
6. [*FileStream*](https://msdn.microsoft.com/ru-ru/library/system.io.filestream%28v=vs.100%29.aspx) поддерживает произвольный доступ к файлам с помощью метода [*Seek*](https://msdn.microsoft.com/ru-ru/library/system.io.filestream.seek%28v=vs.100%29.aspx). По умолчанию класс *FileStream* открывает файлы синхронно, но поддерживает и асинхронные операции. Класс *File* содержит статические методы, а класс *FileInfo* содержит методы экземпляра класса.
7. [*FileSystemInfo*](https://msdn.microsoft.com/ru-ru/library/system.io.filesysteminfo%28v=vs.100%29.aspx) является абстрактным базовым классом для *FileInfo* и *DirectoryInfo*.
8. [*Path*](https://msdn.microsoft.com/ru-ru/library/system.io.path%28v=vs.100%29.aspx) предоставляет методы и свойства для обработки строк каталогов межплатформенным способом.
9. [*DeflateStream*](https://msdn.microsoft.com/ru-ru/library/system.io.compression.deflatestream%28v=vs.100%29.aspx) предоставляет методы и свойства для сжатия и распаковки потоков с использованием *Deflate* алгоритма.
10. [*GZipStream*](https://msdn.microsoft.com/ru-ru/library/system.io.compression.gzipstream%28v=vs.100%29.aspx) предоставляет методы и свойства для сжатия и распаковки потоков. По умолчанию этот класс использует тот же алгоритм, что и класс [*DeflateStream*](https://msdn.microsoft.com/ru-ru/library/system.io.compression.deflatestream%28v=vs.100%29.aspx), но он не может быть расширен для использования других форматов сжатия.
11. [*SerialPort*](https://msdn.microsoft.com/ru-ru/library/system.io.ports.serialport%28v=vs.100%29.aspx) предоставляет методы и свойства для управления файлом ресурсов порта с последовательным выводом данных.
12. Класс *File, FileInfo,* [*DriveInfo*](https://msdn.microsoft.com/ru-ru/library/system.io.driveinfo%28v=vs.100%29.aspx)*, Path, Directory*, и *DirectoryInfo* являются изолированными. Можно создавать новые экземпляры этих классов, но они не могут иметь производных классов.

Данные для отправки сортируются на этапе сравнения. В качестве отправляемых данных используются имена файлов, которые обозначают участок, на котором замечены изменения. По причине использования разных протоколов передачи данных между БЛПА и Землей необходима настройка представления данных для отправки.

## **3.4 Расчет адаптера бортовой системы питания**

По причине того, что модуль основан на одноплатном компьютере необходимо учесть разность в необходимом и имеющемся напряжении питания. Питание одноплатных компьютеров по большей мере имеет требование, как типичное USB-устройство. По спецификации это: *U*пит = 5 В, *I* ≥ 700 мА. Бортовые системы питания имеют напряжение сети 9..36 В. Соответственно необходимо иметь DC-DC преобразователь, который будет учитывать эти характеристики. Для этого подойдет типовое подключение микросхемы 34063 с рассчитанными характеристиками для минимального возможного напряжения [22].

Для типового включения необходимо рассчитать следующие элементы:

1. L1 - дроссель.
2. С1 - времязадающий конденсатор, он определяет частоту преобразования. Максимальная частота преобразования для микросхем 34063 составляет порядка 100 кГц.
3. R2, R1 - делитель напряжения для схемы компаратора. На неинвертирующий вход компаратора подается напряжение 1,25 В от внутреннего регулятора, а на инвертирующий вход - с делителя напряжения. Когда напряжение с делителя становится равным напряжению от внутреннего регулятора - компаратор переключает выходной транзистор.
4. C2, С3 - соответственно, выходной и входной фильтры. Емкость выходного фильтра определяет величину пульсаций выходного напряжения. Если в процессе расчетов получается, что для заданной величины пульсаций требуется очень большая емкость, можно расчет сделать для больших пульсаций, а потом использовать дополнительный LC-фильтр. Емкость С3 обычно берут 100 ... 470 мкФ.
5. Rsc - токочувствительный резистор. Он нужен для схемы ограничения тока. Максимальный ток выходного транзистора для MC34063 = 1.5А, для AP34063 = 1.6А. Если пиковый переключаемый ток будет превышать эти значения, то микросхема может сгореть. Если точно известно, что пиковый ток даже близко не подходит к максимальным значениям, то этот резистор можно не ставить.

Порядок расчета:

1. Выбираем максимальное входное и номинальное выходное напряжения: Vin, Vout и максимальный выходной ток Iout.

*Iout* = 0.7 А, *Vout*= 5 В, *Vin* = 36 В.

1. Выбирают минимальное входное напряжение *Vin(min)*и минимальную рабочую частоту *fmin* при выбранных *Vin* и *Iout*.

*Vin(min)*= 9 В, *fmin*= 100 кГц.

1. Рассчитывают значение *(ton+toff)max* по формуле:

|  |  |
| --- | --- |
| , | (3.3) |

где *ton(max)*- максимальное время, когда выходной транзистор открыт,

*toff(max)*- максимальное время, когда выходной транзистор закрыт.

Исходя их заданных значений получаем:

*(ton+toff)max*= 10 мкс.

1. Рассчитывают отношение *ton/toff* по формуле:

|  |  |
| --- | --- |
| , | (3.4) |

где *VF* - падение напряжения на выходном фильтре,

*Vsat* - падение напряжения на выходном транзисторе (когда он находится в полностью открытом состоянии) при заданном токе.

*Vsat* - определяется по графикам, приведенным в документации на микросхему (или на транзистор, если схема с внешним транзистором).

Из формулы (3.4) видно, что чем больше *Vin*, *Vout* и чем больше они отличаются друг от друга - тем меньшее влияние на конечный результат оказывают *VF* и *Vsat*. Для *Vout*=5 В берется *VF*=0, *Vsat*=1,2 В.

1. Зная *ton/toff* и *(ton+toff)max* решают систему уравнений и находят *ton(max)*.

*ton(max)*= 6.4103 мкс.

1. С1 высчитывается по формуле:

|  |  |
| --- | --- |
| , | (3.5) |

Получаем:

*C1* = 0.28846 нФ.

1. Находят пиковый ток через выходной транзистор:

|  |  |
| --- | --- |
| , | (3.6) |

Получаем:

*IPK(switch)* = 1,4 А.

1. *Rsc* рассчитывают по формуле:

|  |  |
| --- | --- |
| , | (3.7) |

Получаем:

*Rsc*= 0.21429 Ом

1. Рассчитываем минимальную емкость конденсатора выходного фильтра:

|  |  |
| --- | --- |
| , | (3.8) |

где *Vripple(p-p)* - максимальная величина пульсаций выходного напряжения.

*С2* = 8.7500 мкФ.

1. Рассчитываем минимальную индуктивность дросселя:

|  |  |
| --- | --- |
| , | (3.9) |

Получаем:

*L1(min)* =12.8205 мкГн.

1. Сопротивления делителя рассчитываются из соотношения:

|  |  |
| --- | --- |
| , | (3.10) |

Получаем:

*R2/R1* = 3.

Результатом расчета является преобразователь питания, который подходит для большинства систем бортового питания беспилотных летательных аппаратов. Данный адаптер позволяет сохранить значительную часть заряда батареи устройства регистрации данных и при этом является предохранителем для разрабатываемого модуля, в случае возникновения проблем с питанием в бортовой системе.

**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.

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

Класс отображения изменений выполняет функцию показа измененного 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.

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.

## **4.3 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](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B0%D1%81%D0%BD%D1%8B%D0%B9_%D1%86%D0%B2%D0%B5%D1%82), [зеленый](http://ru.wikipedia.org/wiki/%D0%97%D0%B5%D0%BB%D1%91%D0%BD%D1%8B%D0%B9_%D1%86%D0%B2%D0%B5%D1%82), [синий](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D0%B8%D0%B9_%D1%86%D0%B2%D0%B5%D1%82)) — [аддитивная](http://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B4%D0%B8%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D1%81%D0%BC%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%D1%86%D0%B2%D0%B5%D1%82%D0%BE%D0%B2) [цветовая модель](http://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C), как правило, описывающая способ синтеза [цвета](http://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82) для [цветовоспроизведения](http://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82%D0%BE%D0%B2%D0%BE%D1%81%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5). [Аддитивной](http://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B4%D0%B8%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D1%81%D0%BC%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%D1%86%D0%B2%D0%B5%D1%82%D0%BE%D0%B2) она называется потому, что цвета получаются путем добавления к черному.

HSV ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Hue, Saturation, Value — тон, насыщенность, значение) или HSB ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Hue, Saturation, Brightness — тон, насыщенность, яркость) — [цветовая модель](http://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C), в которой [координатами](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BE%D1%80%D0%B4%D0%B8%D0%BD%D0%B0%D1%82%D0%B0) [цвета](http://ru.wikipedia.org/wiki/%D0%A6%D0%B2%D0%B5%D1%82) являются:

* Hue — цветовой [тон](http://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%BD_(%D1%86%D0%B2%D0%B5%D1%82)), (например, [красный](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B0%D1%81%D0%BD%D1%8B%D0%B9_%D1%86%D0%B2%D0%B5%D1%82), [зеленый](http://ru.wikipedia.org/wiki/%D0%97%D0%B5%D0%BB%D1%91%D0%BD%D1%8B%D0%B9_%D1%86%D0%B2%D0%B5%D1%82) или сине-голубой). Варьируется в пределах 0—360°, однако иногда приводится к диапазону 0—100 или 0—1. В [Windows](http://ru.wikipedia.org/wiki/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](http://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D1%8B%D1%89%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C_(%D1%86%D0%B2%D0%B5%D1%82)). Варьируется в пределах 0—100 или 0—1. Чем больше этот параметр, тем «чище» цвет, поэтому этот параметр иногда называют [чистотой цвета](http://ru.wikipedia.org/w/index.php?title=%D0%A7%D0%B8%D1%81%D1%82%D0%BE%D1%82%D0%B0_%D1%86%D0%B2%D0%B5%D1%82%D0%B0&action=edit&redlink=1). А чем ближе этот параметр к [нулю](http://ru.wikipedia.org/wiki/0_(%D1%87%D0%B8%D1%81%D0%BB%D0%BE)), тем ближе цвет к нейтральному [серому](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D1%8B%D0%B9_%D1%86%D0%B2%D0%B5%D1%82).
* Value (значение цвета) или Brightness — яркость. Также задается в пределах 0—100 или 0—1.

Диаграмма классов реализованного модуля определения движения представлена на рисунке 4.1.



Рисунок 4.1 – 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.

## **4.4 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 4.2.



Рисунок 4.2 – 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 приведен в приложении Б.

# **5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ**

**5.1 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).

После принятия команды “Старт” запустится процесс обработки изображения и классификации. Спустя время сформируется результат работы программы с изображением участка с выделенными поражениями и выдан результат о состоянии участка. 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.

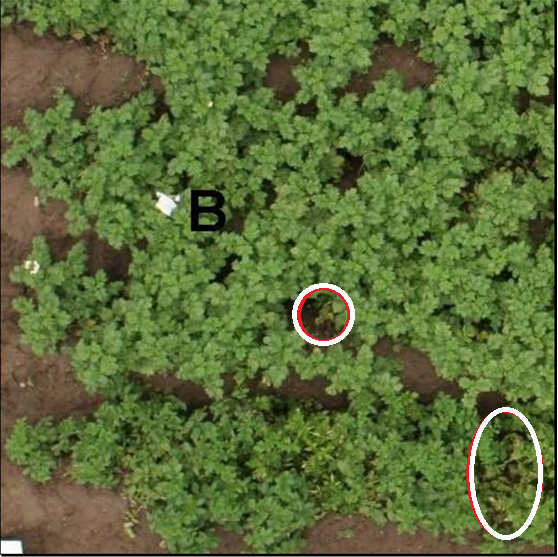


Рисунок 5.1 – Изображения с пораженными участками

После обработки изображения эти области будут выделены на общем фоне. Они имеют яркий окрас для лучшего зрительного восприятия. Так же в графе «Поражения» появляется информация о текущем изменении. Успешная отработка основных задач говорит о правильно выбранных функциях и разработки алгоритмов в конкретной задаче – определении пораженных участков на изображении растительности и ее классификация.

Если в процессе работы программы поменять значения уровня и глубины, то результаты преобразования получатся менее точными. 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

Таким образом, в ходе тестирования программного средства для корректировки карт состояния сельскохозяйственной растительности не было замечено аварийных ситуаций, программа справилась с поставленной задачей выделения областей и регистрации изменений. 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. В качестве одноплатного компьютера был выбран Raspberry Pi 2 в виду распространенности и поддержки со стороны производителя.

Для запуска программы необходим одноплатный компьютер со следующими характеристиками:

- операционная система Windows 10 IoT для Raspberry Pi 2 или Linux Ubuntu;

- процессор с тактовой частотой не ниже 500 МГц;

- оперативная память не меньше 256 Мб;

- объем дискового пространства для программы со всеми модулями 1 Гб.

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

Программное средство было испытано на ЭВМ, работающей на операционных системах Windows 7, Linux Ubuntu 14.04, процессором с тактовой частотой 1.8 Гц, объемом оперативной памяти 2.5 Гб, свободное место на жестком диске 10 Гб, из них всей системой занято 1 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.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Open Source Computer Vision Library*, библиотека компьютерного зрения с открытым исходным кодом) — библиотека алгоритмов [компьютерного зрения](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D0%B7%D1%80%D0%B5%D0%BD%D0%B8%D0%B5), [обработки изображений](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) и численных алгоритмов общего назначения с [открытым кодом](http://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Она содержит все необходимые для работы библиотеки. В процессе написания дипломного проекта была использована библиотека OpenCV 2.4.9. Папку с библиотекой необходимо помещать в корень диска С для Windows 10 или в корневую папку для системы Linux.

Также для функционирования программы необходимо в папку system32 положить файл tbb\_debug.dll. TBB - это кроссплатформенная библиотека шаблонов С++, разработанная компанией Intel для параллельного программирования и содержащая алгоритмы и структуры данных, позволяющие избежать сложностей, возникающих при использовании традиционных реализаций потоков.

## **6.2 Руководство к установке операционной системы на ОК Raspberry Pi**

**6.2.1** Raspberry Pi -  подготовка карты памяти. Так как у Raspberry Pi нет встроенной памяти, для работы компьютера предварительно необходимо  подготовить карту памяти  -  распаковать на нее образ желаемой операционной системы.  Нам понадобится:

* SD (MMC/SDIO) карта памяти объемом от 2 до 32 Гб (скорость считывания должна быть достаточно  высокой, поэтому подойдут карты 6 и 10 класса);
* картридер для подключения карты памяти  к компьютеру;
* программа Win32DiskImager (скачать Win32DiskImager с MEGA  -  ссылка);
* образ операционной системы Linux Ubuntu, который можно скачать с официального сайта бесплатно;
* разархивируем скачанный образ операционной системы  -  для дальнейшей работы нам нужен .img файл;
* подключаем карту памяти с помощью картридера к компьютеру;
* разархивируем программу Win32DiskImager и из папки с программой запускаем файл Win32DiskImager.exe;
* указываем путь к .img файлу с операционной системой (для этого кликаем по изображению папки);
* далее в поле «Device» выбираем из списка носителей карту памяти и нажимаем «Write», после в появившемся диалоговом окне выбираем «Yes» и когда все готово, нажимаем «Write»
* ждем, пока образ запишется на нашу карту памяти, процесс который занимает около 5 минут (Рисунок 6.1.).

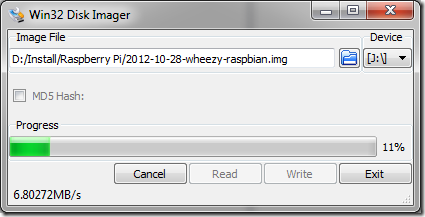


Рисунок 6.1 – Окно записи образа операционной системы на карту

**6.2.2** Raspberry Pi - первое подключение.

После подготовки карты памяти можно приступить к первому запуску  Raspberry Pi.  Для первого включения Raspberry Pi нам понадобится:

* сам компьютер Raspberry Pi;
* карта памяти, с записанной на нее операционной системой;
* USB - клавиатура и USB - мышь;
* монитор с HDMI выходом (и кабель для подключения к нему) или TV тюнер с RSA разъемом (и  кабель для подключения к нему);
* блок питания (5 вольт, минимум 0,5 ампер) с MicroUSB выходом или MicroUSB переходник для  подключения к компьютеру.

Подключаем в следующем порядке: карту памяти вставляем в разъем на Raspberry Pi, к USB -  разъемам подключаем клавиатуру и мышь, кабелем HDMI (или RSA) подключаем к Raspberry Pi монитор, и подключаем питание через MicroUSB -  разъем. После подключения питания сразу начнется загрузка операционной системы.  После появляется окно настроек, которое появляется при первом запуске операционной системы: Меню настроек Raspberry Pi (рисунок 6.2).

Рассмотрим пункты меню, которыми будем пользоваться для настройки:

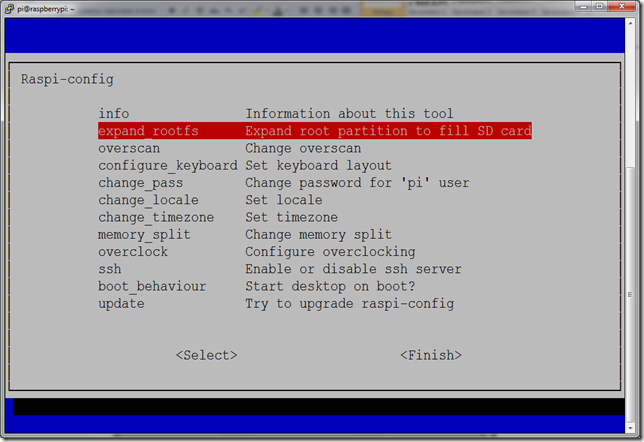


Рисунок 6.2 – Меню настроек Raspberry Pi

1. *expand\_rootfs* - здесь можно увеличить *root* размер на весь размер карты памяти. Рекомендуется это сделать в первую очередь.
2. *configure\_keyboard* - в этом пункте вы можете выбрать драйвер клавиатуры.
3. *change\_pass* - смена пароля пользователя "*pi*". При вводе пароль не отображается. Ввести необходимо дважды.
4. *change\_locate* - установка языка системы (рисунок 6.2). Выбирать лучше вариант русской локализации в кодировке UTF-8. Выбор осуществляется пробелом (рисунок 6.3).

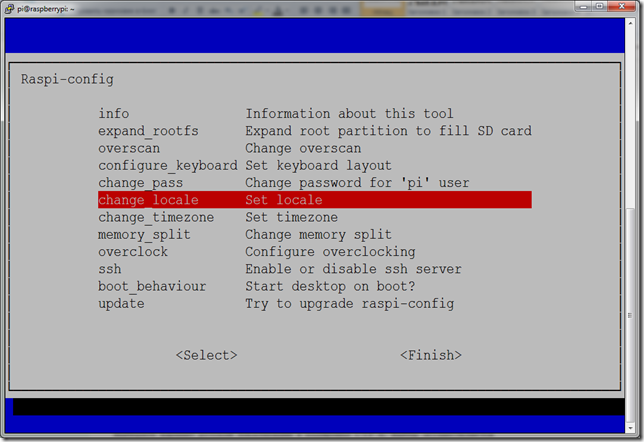


Рисунок 6.2 –Пункт *change\_locate* в меню настроек Raspberry Pi

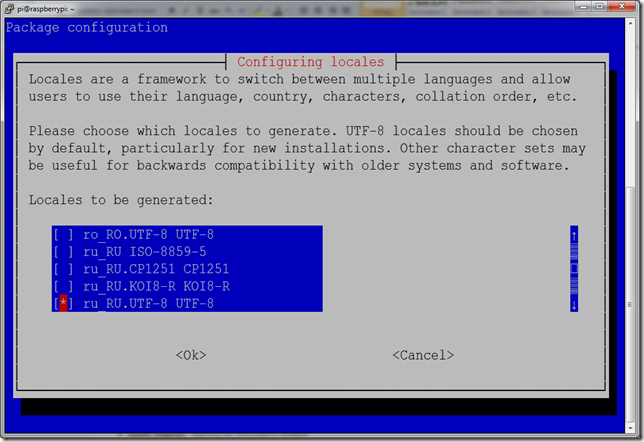


Рисунок 6.3 – Выбор локализации

1. *memory\_split* - распределение памяти Raspberry Pi (рисунок 6.34). Необходимо определиться, сколько оперативной памяти лучше выделить для графического процессора. При работе в консоли будет достаточно и 16 Мб, для просмотра видео в графической оболочке - 64-128 Мб. Выбранные значения могут быть только: 16, 32, 64, 128 или 256 (рисунок 6.5).

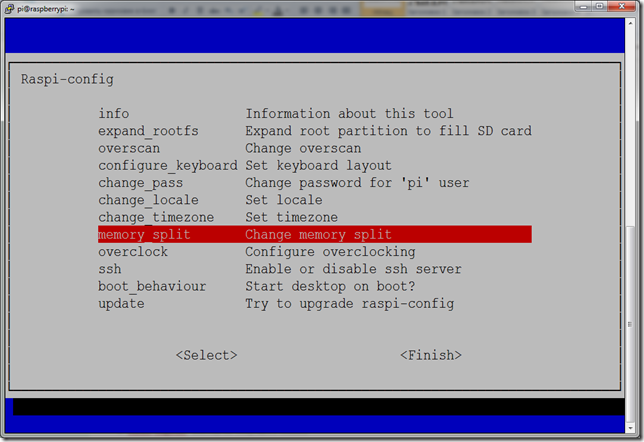


Рисунок 6.4 –Пункт *memory\_split* в меню настроек Raspberry Pi



Рисунок 6.5 – Выбор объема памяти

1. *overclock* - разгон процессора Raspberry Pi (рисунок 6.6). Оставляем по умолчанию. В случае необходимости выбирается подходящий вариант. Здесь следует учесть тот факт, что при увеличении частоты, также увеличивается и потребление энергии. То есть источник питания должен выдавать достаточный ток для нормальной работы (рисунок 6.7).

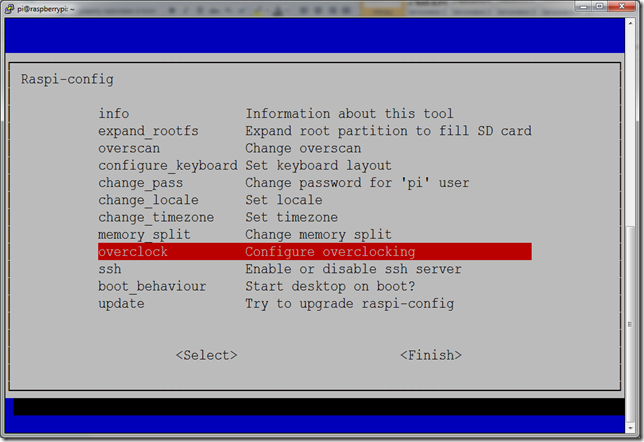


Рисунок 6.6 –Пункт *overclock* в меню настроек Raspberry Pi

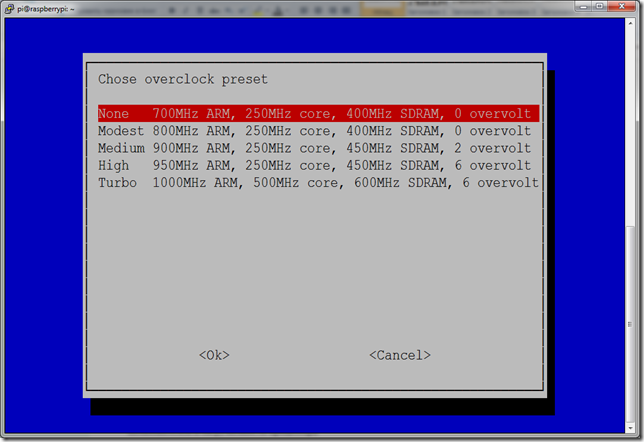


Рисунок 6.7 – Частоты процессора Raspberry Pi

1. *ssh* - включение или выключение SSH сервера (рисунок 6.8). Рекомендуется включить SSH для возможности использовать удаленное управление.

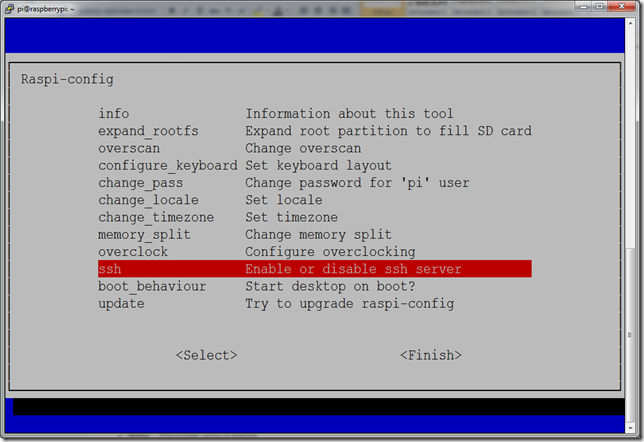


Рисунок 6.8 –Пункт *ssh* в меню настроек Raspberry Pi

1. *boot\_behaviour* - загрузка визуальной оболочки при загрузке системы (рисунок 6.9). Для загрузки визуальной оболочки используется команда *startx*.

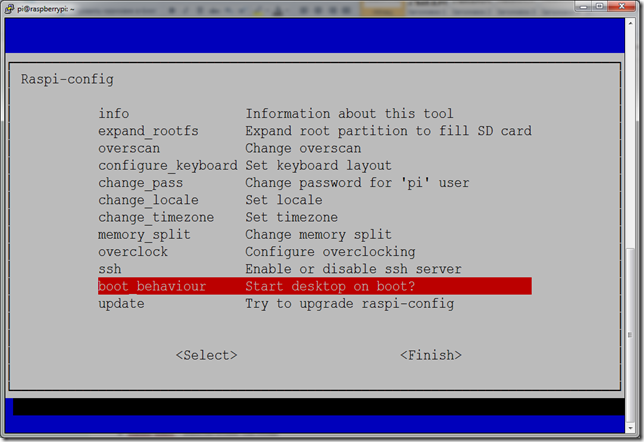


Рисунок 6.9 –Пункт *boot\_behaviour* в меню настроек Raspberry Pi

После завершения настроек нажмите на клавиатуре Ctrl+F, выбираем <Finish>. Raspberry Pi уйдет на перезагрузку для внесения изменений.

Для установки пароля пользователю "root" в Raspberry Pi нужно набрать в консоли команду sudo passwd root и ввести пароль дважды.

# **7 РАСЧЕТ ТЕХНИКО-ЭКОНОМИЧЕСКОГО ОБОСНОВАНИЯ ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ ИНВЕСТИЦИОННОЙ ОЦЕНКИ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ ИСПОЛЬЗОВАНИЯ РАЗРАБОТКИ И ВНЕДРЕНИЯ АППАРАТНО-ПРОГРАММНОГО МОДУЛЯ КОРРЕКТИРОВКИ КАРТЫ СОСТОЯНИЯ РАСТИТЕЛЬНОСТИ ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ РЕГИСТРАЦИИ ДАННЫХ**

## **7.1 Характеристика разрабатываемой системы**

Аппаратно-программный модуль корректировки карты состояния растительности для мобильных устройств регистрации данных позволяет проводить автоматизированную экспресс оценку состояния растительности и упрощает работу заказчика. Автономность позволяет предупреждать болезни растений и потерю урожая на раннем этапе ее развития.

## **7.2 Расчет себестоимости и отпускной цены единицы продукции**

В нашем случае в себестоимость продукции включим следующие статьи затрат: «Основная заработная плата производственных рабочих», , «Общехозяйственные расходы».

По статьям «Сырье и материалы за вычетом возвратных отходов», «Покупные комплектующие изделия, полуфабрикаты», «Износ инструментов и приспособлений целевого назначения», «Общепроизводственные расходы», «Коммерческие расходы», «Прочие производственные расходы» мы не исчисляем себестоимость, так как для разработки программного средства никакие материальные ресурсы не требуются, а остальные статьи расходов не оказывают значительного влияния на себестоимость.

**7.2.1** Расчет затрат по статье «Основная заработная плата производственных рабочих»

Так как программная система является продуктом интеллектуальной деятельности, а не материально-вещественным объектом – считаем, что для ее разработки не понадобятся какие-либо материалы, сырье и покупные комплектующие изделия. Поэтому опускаем следующие статьи по расчету себестоимости: «Сырье и материалы за вычетом возвратных отходов» и «Покупные комплектующие изделия, полуфабрикаты».

Срок разработки программного продукта установлен в 5 месяцев. Система оплаты – сдельно-премиальная.

Сумма основной заработной платы по договору составляет

Зо = 6 500 000 руб.

Система была разработана за 1 месяц до конца поставленного срока, вследствие чего предприятие выплатила программисту Коромыслову А.В. премию в виде дополнительной заработной платы, используя норматив дополнительной заработной платы предприятия, Нд = 20%.

Расчет производится по формуле (6.1):

|  |  |
| --- | --- |
| , | (7.1) |

где Зо – сумма основной заработной платы, Зо = 6 500 000 руб.;

Нд – норматив дополнительной заработной платы предприятия.



Так как Коромыслов справился с работой на 1 месяц раньше, согласно заключенному договору предприятие должно выплатить 6 500 000 руб., а дополнительную заработную плату рассчитать за 4 месяца работы по формуле (7.2):

|  |  |
| --- | --- |
| , | (7.2) |

где *Nm*– фактически отработанное время, *Nm*= 4 мес.;

Зд – дополнительная заработная плата, Зд = 1 300 000 руб.



Сумма общей начисленной заработной платы вычисляется по формуле (7.3):

|  |  |
| --- | --- |
| , | (7.3) |

где Зо – сумма основной заработной платы, Зо = 6 500 000 руб.;

Здобщ – дополнительная заработная плата, Здобщ = 5 200 000 руб.



**7.2.2** Расчет затрат по статье «Отчисления в фонд социальной защиты населения»

Сумму отчислений в фонд социальной защиты вычислим по формуле (7.4):

|  |  |
| --- | --- |
| , | (7.4) |

где Зобщ – общая заработная плата, Зобщ = 11 700 000 руб.;

Нно – ставка отчислений в фонд социальной защиты населения, установленная законодательством, Нно = 34%.



**7.2.3** Расчет затрат по статье «Отчисления по обязательному страхованию»

Рассчитаем выплаты единого налога от фонда оплаты труда по формуле (7.5):

|  |  |
| --- | --- |
| , | (7.5) |

где Зобщ – общая заработная плата, Зобщ = 11 700 000 руб.;

Нстр – ставка налога отчисления по обязательному страхования,

Нстр = 1%.



**7.2.4** Расчет затрат по статье «Общехозяйственные расходы»

Для работы специалиста на рабочем месте были понесены некоторые хозяйственные расходы (расходы на канцелярские товары, оплата за электроэнергию). Рассчитаем эти расходы по формуле (7.6):

|  |  |
| --- | --- |
| , | (7.6) |

где Зо – сумма основанной заработной платы, Зо = 6 500 000 руб.;

Нопх – норматив общехозяйственных расходов, Нопх = 70%



**7.2.5** Расчет полной себестоимости

Рассчитаем полную себестоимость по формуле (7.7):

|  |  |
| --- | --- |
| , | (7.7) |

где Зо – общая заработная плата основных производственных рабочих,

Зо = 6 500 000 руб.;

Зд – дополнительная заработная плата основных производственных рабочих, Зд = 1 300 000 руб.;

Рсоц – отчисления в фонд социальной защиты, Рсоц = 3 978 000 руб.;

Рстр – отчисления по обязательному страхованию,

Рстр = 117 000 руб.;

Ропх – общехозяйственные расходы, Ропх = 4 550 000 руб.



Таблица 7.1 – Калькуляция полной себестоимости

|  |  |  |
| --- | --- | --- |
| Наименование статьи затрат | Условное обозначение | Сумма, руб. |
| Основная заработная плата основных производственных рабочих | Зо | 6 500 000 |
| Дополнительная заработная плата основных производственных рабочих | Зд | 1 300 000 |
| Отчисления в фонд социальной защиты населения | Рсоц | 3 978 000 |
| Отчисления по обязательному страхованию | Рстр | 117 000 |
| Общехозяйственные расходы | Ропх | 4 550 000 |
| Производственная себестоимость | Спр | 16 445 000 |
| Полная себестоимость | Сп | 16 445 000 |

Для наглядности приведем круговую диаграмму калькуляции полной себестоимости, представленную на рисунке 7.1.

Рисунок 7.1 – Круговая диаграмма калькуляции полной себестоимости

## **7.3 Расчет интегрированного экономического эффекта в сфере потребления новой техники**

**7.3.1** Расчет показателей экономической эффективности проекта

При оценке эффективности инвестиционных проектов необходимо осуществить приведение затрат и результатов, полученных в разные периоды времени, к расчетному году, путем умножения затрат и результатов на коэффициент дисконтирования αt, который определяется по формуле (7.8):

|  |  |
| --- | --- |
| , | (7.8) |

где Ен – требуемая норма дисконта, Ен = 15%;

*t* – порядковый номер года, затраты и результаты которого приводятся к расчетному году;

*tp* – расчетный год, в качестве расчетного года принимается год вложения инвестиций, *tp* = 1.

;

;

;

;

Интегрированный экономический эффект рассчитывается по формуле (7.9):

|  |  |
| --- | --- |
| , | (7.9) |

где ЧДДt – чистый дисконтированный доход за год t, млн. руб.



Таблица 7.2 – Расчет экономического эффекта от использования нового ПС

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Наименование показателя | Ед. изм. | Усл. обозн. | по годам производства | | | |
| 2016 | 2017 | 2018 | 2019 |
| Результат с учетом фактора времени | млн. руб. | Р*t*α*t* | 0,71 | 6,3 | 5 | 3,5 |
| Инвестиционные вложения | млн. руб. | З*t* | 11,5 | - | - | - |
| Затраты с учетом фактора времени | млн. руб. | З*t*α*t* | 11,5 | - | - | - |
| Чистый дисконтированный доход | млн. руб. | ЧДД | -10,8 | 5,8 | 4,5 | 3,2 |
| Чистый дисконтированный доход с нарастающим итогом | млн. руб. | Эинт | -10,8 | -5 | -0,5 | 2,7 |
| Коэффициент дисконтирования |  | α*t* | 1 | 0,87 | 0,76 | 0,66 |

Как видно из таблицы чистый дисконтированный доход имеет максимальное значение во втором году и составляет 5,8 млн. руб., а интегрированный экономический эффект за четыре года реализации проекта составил 2,7 млн. руб.

На рисунке 7.2 отражена диаграмма с чистым дисконтированным доходом и интегрированным экономическим эффектом за 4 года.

Рисунок 7.2 – Чистый дисконтированный доход и интегрированный экономический эффект за 4 года

**7.3.2** Расчет срока окупаемости инвестиций

Расчет срока окупаемости инвестиций производится по формуле (7.10):

|  |  |
| --- | --- |
| , | (7.10) |

где P*t* – чистый доход, полученный в году *t*, руб.;

З*t* – затраты (инвестиции) в году *t*, руб.;

α*t* – коэффициент дисконтирования в году *t*, %.

















Из расчетов пункта 7.3.2 видно, что срок окупаемости инвестиций Ток = 4 года.

В процессе расчета технико-экономического обоснования экономической эффективности инвестиционной оценки технологических процессов использования разработки и внедрения аппаратно-программного модуля для корректировки карт состояния растительности на мобильных устройствах регистрации данных были получены следующие результаты, исходя из таблиц 7.1 и 7.2:

* чистый дисконтированный доход имеет максимальное значение во втором году и составляет ЧДД = 5,8 млн. руб.;
* интегрированный экономический эффект за четыре года реализации проекта составил Эинт= 2,7 млн. руб.;
* срок окупаемости инвестиций составит Ток = 4 года;
* полная себестоимость составит в сумме равной Сп= 16,445 млн. руб.

Таким образом, использование разработки и внедрения аппаратно-программного модуля для корректировки карт состояния растительности на мобильных устройствах регистрации данных, свидетельствует о целесообразности вложения инвестиций, поскольку является эффективной для коммерческого успеха.

# **Заключение**

В ходе выполнения дипломного проекта был разработан и реализован аппаратно-программный модуль корректировки карты состояния растительности для мобильных устройств регистрации данных. Данный модуль 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 программная часть модуля;
* реализована система классификации изображений;
* произведены экономические расчеты, обоснована рентабельность системы;
* подведены итоги по проделанной работе;
* подготовлена пояснительная записка и графический материал по теме дипломного проекта.

Модуль является упрощенным аналогом современных систем из области обработки и анализа изображений в точном земледелии и может быть успешно внедрена в производство, а так же в сферы научной деятельности. В ходе выполнения испытаний модуль показал свою работопригодность для использование как самостоятельного решения.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Greenbelarus [Электронный ресурс]. – Режим доступа: <http://greenbelarus.info/files/downloads/zashchita_rasteniy_v_organicheskom_selskom_hozyaystve.pdf>

[2] Якушев, М. В., Информационное обеспечение точного земледелия / М. В. Якушев, Якушев В.В. – СПб.: Питер Ком, 2007. – 312 с.

[3] Теория картографии [Электронный ресурс]. – Режим доступа: <http://yug-gps.narod.ru/docs/002x/st027.htm>

[4] Sibgeomap [Электронный ресурс]. – Режим доступа: <http://www.sibgeomap.com/technology/oddz>

[5] Nsu [Электронный ресурс]. – Режим доступа:<http://www.nsu.ru/xmlui/bitstream/handle/nsu/318/03.pdf;jsessionid=F6A19137B3C6CC404E3CF081A2FEDF8F?sequence=1#page=2&zoom=auto,-202,556>

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

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

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

[9] Ганченко, В. В. Алгоритмы распознавания площадных объектов на аэрофотоснимках сельскохозяйственных полей на основе комбинирования яркостных и текстурных признаков (автореф. дис. на соискание ученой степени канд. тех. наук : 05.13.01 / В. В. Ганченко ; науч. рук. А. А. Дудкин ; ОИПИ. - Минск, 2014. - 174 с.

[10] Atb [Электронный ресурс]. – Режим доступа: <http://www.atb-potsdam.de/uploads/feupload/Gebbers_Keynote_9ECPA_2013_Lleida.pdf>

[11] ВНИИЗ [Электронный ресурс]. – Режим доступа: <http://vniiz.org/article.aspx?Id=60>

[12] EcoRasum [Электронный ресурс]. – Режим доступа: <http://eco-razum.com/?q=node/37>

[13] EcoRasum [Электронный ресурс]. – Режим доступа: <http://eco-razum.com/?q=node/48>

[14] Trimble [Электронный ресурс]. – Режим доступа: [www.trimblegnss.ru](http://www.trimblegnss.ru).

[15] Беляев, Б. И. Оптическое дистанционное зондирование : монография / Б. И. Беляев, Л. В. Катовский. – Минск : БГУ, 2006. – 455 с.

[16] Chem-astu [Электронный ресурс]. – Режим доступа http://www.chem-astu.ru

[17] Wiki [Электронный ресурс]. – Режим доступа https://ru.wikipedia.org/wiki/%D0%9F%D0%9B%D0%98%D0%A1

[18] Wiki [Электронный ресурс]. – Режим доступа https://dl.dropboxusercontent.com/u/4035896/a320\_downloads/SBC\_comparison44.pdf

[19] On-Board Vision Processing For Small UAVs [Электронный ресурс]. – Режим доступа : <http://arxiv.org/ftp/arxiv/papers/1504/1504.07021.pdf>

[20] Zastosowanie technik analizy obrazu do wczesnego wykrywania patogeno`w ziemniaka. Praca nie publicowana / B. Sobkowiak [et al.]. – Poznan`: PIMR, 2006.

[21] Zastosowanie technik analizy obrazu do wczesnego wykrywania zarazy ziemnechanej w warynkach polowych. Praca nie publicowana / B. Sobkowiak [et al.]. – Poznan`: PIMR, 2007.

[22] Datasheet [Электронный ресурс]. – Режим доступа https://www.sparkfun.com/datasheets/IC/MC34063A.pdf

[23] Datasheet [Электронный ресурс]. – Режим доступа <http://dragon.larc.nasa.gov/background/pubabs/papers/msrcrcmpr.pdf>

# **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 создает 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>

/// <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">Текуцая позиция по Х</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);

}

}

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);

}

}

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);

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));

}