Санкт-Петербургский политехнический университет Петра Великого

Институт прикладной математики и механики

Кафедра прикладной математики

Документация программного комплекса

*«Там, где море»*

Выполнили: студенты групп 13641/1,3,4

|  |
| --- |
| Лысикова Наталья |
| Зефиров Артем |
| Пестова Наталья |
| Выменец Максим |
| Тхакушинова Рузанна |

Санкт-Петербург

2017

**Аннотация**

Настоящий документ является описанием программного комплекса для вычислений характеристик движений объектов, наблюдаемых при помощи радиолокатора. Указанный комплекс разработан в рамках учебного курса «Эффективная разработка программного обеспечения», проводимого рамках магистерской программы по направлению «прикладная математика» в Санкт-Петербургском политехническом университете им. Петра Великого и предназначен для использования в учебных целях.

Настоящий документ содержит необходимые сведения о назначении, основных характеристиках, входных и выходных данных, логической структуре программного комплекса.

**Оглавление**

[Словарь терминов 5](#_Toc484249248)

[1. Общие сведения 6](#_Toc484249249)

[1.1. Наименование и цель программного комплекса 6](#_Toc484249250)

[1.2. ПО, необходимое для функционирования программы 6](#_Toc484249251)

[1.3. Языки программирования, на которых написана программа 6](#_Toc484249252)

[2. Функциональное назначение 7](#_Toc484249253)

[2.1. Решаемые задачи 7](#_Toc484249254)

[2.2. Функциональные и количественные ограничения 7](#_Toc484249255)

[3. Описание логической структуры 8](#_Toc484249256)

[3.1. Общая структура программы 8](#_Toc484249257)

[3.2. Метод работы 9](#_Toc484249258)

[3.3. Описание используемых структур данных 10](#_Toc484249259)

[3.4. Описание используемых методов 11](#_Toc484249260)

[3.5. Алгоритмы 15](#_Toc484249261)

[3.5.1. Алгоритмы, используемые в библиотеке Radar.lib 15](#_Toc484249262)

[2.1.1. Алгоритмы, используемые в работе компонента GUI 17](#_Toc484249263)

[3. Вызов и загрузка 20](#_Toc484249264)

[4. Входные данные 27](#_Toc484249265)

[5. Выходные данные 29](#_Toc484249266)

# Словарь терминов

Радарное изображение – цифровое изображение круга, содержащее на нейтральном фоне контуры объектов и цифровой шум, получаемое в результате радиолокационных наблюдений

Цифровой шум – дефект радарного изображения

Объект на изображении – набор пикселей радарного изображения, соответствующий наблюдаемому физическому объекту

Относительная система координат – декартовая система координат, центром которой является центр радарного изображения, с направлением осей абсцисс и ординат в соответствии с горизонтальным и вертикальным направлением на радарном изображении.

Координаты объекта – среднее арифметическое координат его пикселей в относительной системе координат

Скорость объекта – скорость физического объекта в относительной системе координат

Размер объекта – длина и ширина физического объекта, соответствующая занимаемому на радарном изображении количеству пикселей

Характеристики движения объекта – координаты и скорость объекта

# Общие сведения

## Наименование и цель программного комплекса

Программный комплекс «Там, где море» предназначен для обработки радарных изображений и вычисления характеристик движения и физических параметров идентифицированных на изображениях объектов.

## ПО, необходимое для функционирования программы

Программный комплекс предназначен для программных платформ, имеющих возможность выполнения кода, написанного на языке C++. Это включает:

* выполнение на 32- и 64-битных ОС Linux и Windows для ПК и серверов;
* статическую и динамическую линковку с программными комплексами C++ и других компилируемых языках, предоставляющих возможность статической и динамической линковки с программными библиотеками, предоставляющими C++ интерфейс

## Языки программирования, на которых написана программа

Проект реализован на языке C++ с использованием фреймворка Qt5.

# Функциональное назначение

## Решаемые задачи

Программный комплекс «Там, где море» выполняет обработку радарных изображений, поступающих в файловую систему от радиолокатора.

В процессе первичной обработки устраняется цифровой шум, мешающий выявлению физических объектов на изображения. В результате идентификации физических объектов с учетом входной информации о времени наблюдения и масштабе изображения вычисляются характеристики движения объекта и размеры объекта.

## Функциональные и количественные ограничения

Вывод времени и даты осуществляется в шкале UTC.

Координаты и скорости объектов выводятся в относительной системе координат в (км/ч). Размеры объектов выводятся в квадратных метрах.

Масштаб радарного изображения задается параметром «scale» и устанавливается по умолчанию 100 м/пиксель. Данный параметр определяется характеристиками локатора и определяет нижнюю оценку точности, с которой определяются характеристики движения и размер объектов. Значения параметра ограничены диапазоном от 1 до 10000.

# Описание логической структуры

## Общая структура программы

Программный комплекс «Там, где море» состоит из следующих компонент:

1. Исполняемый модуль *Gui*, предоставляющий пользователю интерфейс взаимодействия с программой и обеспечивающего связь с остальными компонентами комплекса.
2. Библиотека *Radar.lib,* осуществляющая анализ радарных изображений и формирующая выходной файл программы.
3. Внешняя библиотека *EasyBMP.lib* для работы с файлами формата BMP.

Входными файлами являются файлы формата BMP, содержащие радарные изображения. Выходными – текстовые файлы, содержащие информацию о характеристиках движения объектах и их размерах, а также лог-файл, содержащий сведения о работе программы.

Компоненты программного комплекса отображены на диаграмме размещения (рисунок 1):

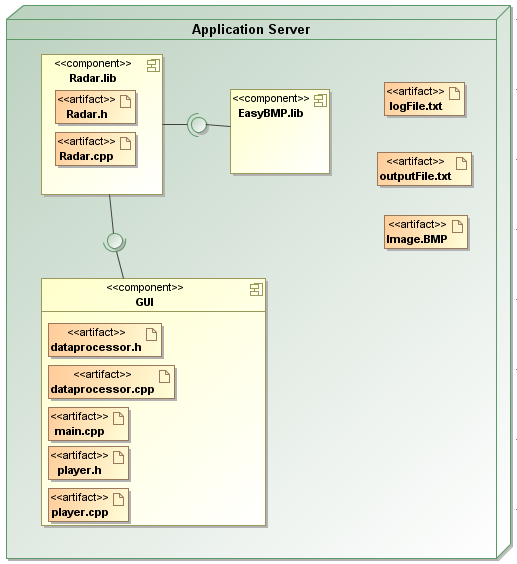


Рисунок 1 – Диаграмма размещения компонент программы

## Метод работы

Работа с библиотекой *Radar.lib* осуществляется по следующей схеме:

1. Инициализация объекта класса Radar.
2. Установка настроек, к которым относятся:

* путь к директории с файлами радарных изображений
* путь к выходному файлу
* указатель на лог-файл, созданный в компоненте GUI
* параметр минимального количества файлов для обработки
* параметр масштаба радарного изображения.

1. Запуск метода обработки файлов, находящихся в очереди в одном из потоков компонента GUI.
2. Уничтожение объекта класса и освобождение ресурсов.

*Примечание:* все указанные операции относятся к одному сеансу работы с *Radar.lib*. Пользователь имеет возможность работать с библиотекой в нескольких сеансах независимо, в этом случае шаги 1-4 должны выполняться независимо для каждого сеанса. Такой сценарий работы имеет смысл при использовании параллельных алгоритмов обработки радарных изображений.

## Описание используемых структур данных

Класс *Radar* является единственным классом библиотеки *Radar.lib* и осуществляет обработку входных радарных изображений и формирование выходного файла с результатами характеристиками движений объектов и их размеров.

Таблица 1. Поля класса *Radar*

|  |  |
| --- | --- |
| **Поле/Поля** | **Описание** |
| int freq | Минимальное количество файлов, обрабатываемых функцией run(). Значение по умолчанию: 1 |
| int YYYY, MM, DD, hh, mm, ss, cc | Год, месяц, день, час, минута, секунда и сотая часть секунды времени наблюдения |
| int oYYYY, oMM, oDD, ohh, omm, oss | Год, месяц, день, час, минута, секунда и сотая часть секунды времени наблюдения предыдущего изображения |
| int blackAndWhitePixels | Количество пикселей изображения, не являющихся пикселями рамки |
| int clearedPixels | Количество пикселей изображения. изменивших цвет при очистке изображения от шума |
| double scale = 100 | Масштаб изображения в метрах на пиксель |
| double noize | Оценка уровня зашумлённости изображения |
| bool firstImage = true | Переменная, в которой хранится, является ли изображение первым в серии. |
| BMP world | Текущее радарное изображение |
| int minMonitoredSize | Параметр, указывающий, насколько большие объекты не следует отбрасывать как ложные с учётом оценки шума |
| const double maxDelta = 10 | Параметр, указывающий, насколько большой допускается значение метрики delta между объектами на предыдущем и на текущем изображении для признания этих объектов одним реальным. |
| vector<double> ox, oy, os, ovx, ovy | Характеристики объектов, детектированных на предыдущем изображении. |
| vector<bool> ot | Характеристика объекта, указывающая, был ли он замечен на предыдущем изображении |
| const RGBApixel white = { 255,255,255,0 } | Белый цвет пикселя |
| const RGBApixel black = { 0,0,0,0 } | Чёрный цвет пикселя |
| const RGBApixel green = { 0,255,0,0 } | Зелёный цвет пикселя |
| const RGBApixel red = { 0,0,255,0 } | Красный цвет пикселя |
| string logFileName | Имя файла для вывода лога работы программы |
| string outputFileName | Имя файла для вывода результатов работы программы |
| FILE \* logFile | Файл для вывода лога работы программы |
| FILE \* outputFile | Файл для вывода рещультатов работы программы |
| int[4] dx = { 0, 1, 0, -1 } | Перемещения по первой координате при переборе соседних пикселей |
| int[4] dy = { 1, 0, -1, 0 } | Перемещения по второй координате при переборе соседних пикселей |

## Описание используемых методов

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| **Методы видимости private** | |
| bool iw(int x, int y) | Проверка, что пиксель с данными координатами находится в границах изображения и не является пикселем рамки |
| void cleanPixel(int x, int y) | Локальная очистка шума, метод подробнее описан ниже |
| double clean(); | Глобальная очистка шума, возвращающая оценку уровня шума, метод подробнее описан ниже |
| vector<double> fillingPixel(RGBApixel from, RGBApixel to, int x, int y) | Заливка смежной одноцветной части изображения другим цветом, возвращающая количество и сумму координат залитых пикселей, метод подробнее описан ниже |
| void findObjects(); | Поиск объектов на изображении, метод подробнее описан ниже |
| double delta(double x1, double y1, double s1, double x2, double y2, double s2) | Метрика, определяющая непохожесть объектов. Равна 20 \* sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2)) / (s1 + s2) + max(s1,s2)/min(s1,s2) |
| BMP nextStep(BMP image, bool createRedArrows); | Метод, принимающий новое изображение и определяющий положение и скорости объектов на нём, подробнее описан ниже |
| double timeInterval(); | Метод, вычисляющий время, прошедшее между наблюдением предыдущего и текущего радарного изображения |
| void writeOutput(); | Метод, выводящий характеристики объектов в выходной файл |
| **Методы видимости public** | |
| Radar(double newScale) | Конструктор класса, принимает масштаб изображения |
| ~Radar() | Деструктор класса |
| void setFreq(int newFreq) | Метод изменения минимального количества входных изображений |
| void clearCashe(); | Метод, удаляющий информацию о предыдущих наблюдениях |
| void setOutputFile(const string newOutputFileName) | Метод для задания выходного файла |
| void setLogFile(const string newLogFileName) | Метод для задания файла логирования |
| void setLogFile(FILE \* newLogFile) | Метод для задания файла логирования |
| void setScale(int newScale) | Метод для изменения масштаба |
| void setIdentThreshold(double newThreshold) | Пустой метод, необходимый для взаимодействия с интерфейсом |
| int run(const list<string> inputFileNames, bool createOutputImage, const string outputImageFileName) | Метод для приёма новых изображений и вызова методов обработки. Эта версия метода выводит выходные изображения в целях тестирования |
| int run(const list<string> inputFileNames) | Метод для приёма новых изображений и вызова методов обработки. Эта версия метода не выводит выходные изображений |
| BMP getImage() | Метод, возвращающий текущее изображение в целях тестирования |

**Главный метод nextStep:**

1. Вызов метода clean.

2. Вызов метода findObjects.

3. Изменение длины вектора ot на равную количеству объектов и присваивание всем его элементам значения false.

3. Если изображение является первым в серии - перейти к шагу 10.

4. Если изображение не первое, то составить матрицу costs. В элементе costs[i][j] находится значение метрики delta различия i-го объекта, найденного при предыдущем вызове алгоритма (т.е. методом findObjects для предыдущего изображения) и j-го объекта, найденного при этом вызове. Также составить вектора costsI и costsJ, в которых costsI[i] = i и costsJ[j] = j.

5. Найти в матрице costs наименьший элемент costs[minI][minJ]

6. Если costs[minI][minJ] больше константы maxDelta (параметр задается вручную и характеризует ограничение на изменение объекта между радарными изображениями), перейти к шагу 10.

7. Покоординатно вычесть из расположения minJ-го объекта, найденного при этом вызове алгоритма, расположение minI-го объекта, найденного при предыдущем вызове, принять это в качестве скорости объекта по соответствующей координате. (при выводе результатов скорость будет разделена на время, прошедшее между изображениями)

8. Удалить из матрицы costs строку minI и столбец minJ, из вектора costsI - элемент с индексом minI, из вектора costsJ - элемент с индексом minJ. Таким образом, всегда получается, что costsI[i] - изначальный индекс в матрице costs строки, у которой сейчас индекс i, аналогично с столбцами и costsJ.

9. Вернуться к шагу 6.

10. Вернуть текущее изображение (в целях отладки, на практике оно в дальнейшем не используется).

**Метод clean:**

В данном методе происходит очистка изображения от шума.

1. Присвоить переменным blackAndWhitePixels и clearedPixels значение 0.

2. Вызвать метод cleanPixel поочерёдно для всех пикселей изображения.

3. Вернуть частное значений переменных clearedPixels и blackAndWhitePixel.

*Метод сleanPixel:*

В данном методе происходит локальная очистка изображения от шума.

1. Если пиксель изображения, для которого вызван метод, не является белым или чёрным, завершить выполнение метода.

2. Увеличить переменную blackAndWhitePixels на 1.

3. Перебрать соседние пиксели изображения, сохранить в переменную same число совпадающих с данным пикселем по цвету, а в переменную all количество соседних пикселей, не являющихся пикселями рамки.

4. Если same меньше 2, а all больше 1, обратить цвет пикселя, увеличить переменную clearedPixels на 1 и вызвать метод для соседних пикселей.

**Метод findObjects:**

1. Перейти к очередному пикселю изображения, если все пиксели уже перебраны, закончить выполнение метода.

2. Если пиксель не белый, вернуться к шагу 1.

3. Если пиксель белый, вызвать для него метод fillingPixel (см. ниже) с белым цветом в качестве исходного и зелёным в качестве конечного, и сохранить его выходные данные [x,y].

4. Если размер предполагаемого объекта не меньше параметра minMonitoredSize, зависящего от оценки уровня шума на основании статистических данных, то записать его характеристики в список объектов - в качестве каждой координаты следует записать вычисленную методом fillingPixel сумму соответствующих координат, разделённую на размер.

5. Если размер предполагаемого объекта меньше minMonitoredSize, вызвать для текущего пикселя метод fillingPixel с зелёным цветом в качестве исходного и чёрным в качестве конечного.

6. Вернуться к шагу 1.

**Метод fillingPixel:**

Метод вызывается для пикселя изображения p[x,y] и принимает два цвета в качестве исходного и конечного. Метод возвращает количество (оно же размер объекта) и сумму координат (для каждой координаты отдельно) перекрашенных пикселей в виде вектора из трёх значений.

1. Если цвет пикселя не совпадает с текущим, завершить выполнение метода. В качестве всех выходных значений вернуть нули.

2. Заменить цвет пикселя на конечный, рекурсивно вызвать метод для всех соседних пикселей. Вернувшиеся соответствующие значения сложить.

3. К сумме размеров прибавить единицу, к суммам сумм координат прибавить соответствующие координаты текущего пикселя.

4. Вернуть получившиеся значения.

## Алгоритмы

## Алгоритмы, используемые в библиотеке Radar.lib

Для очистки изображения используется вариация метода логического сглаживания/логического усреднения. Для идентификации на очередном изображении детектированных ранее объектов используется вариация приблизительного жадного алгоритма решения задачи о назначениях.

Подробно алгоритмы, использующиеся при обработке изображений, описаны в пункте 3.4.

* + 1. **Алгоритм преобразования координат**

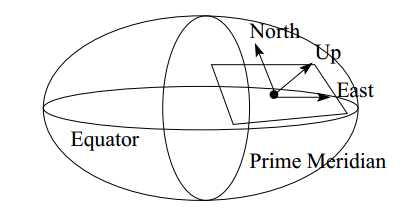
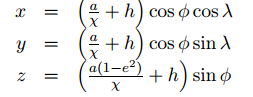


Рисунок 1– Локальная система координат

Переход из географической системы координат к локальной декартовой системе координат производится в несколько этапов:

1. Определить широту , долготу  и высоту над поверхностью эллипсоида 
2. Выразить малые изменения в широте, долготе и высоте в геоцентрической системе координат

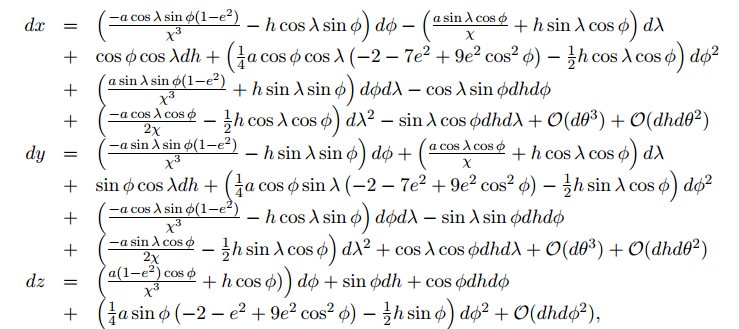
,

где

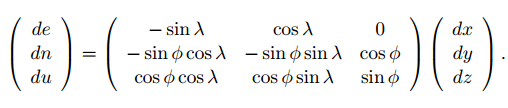


 большая полуось и эксцентриситет эллипсоида.

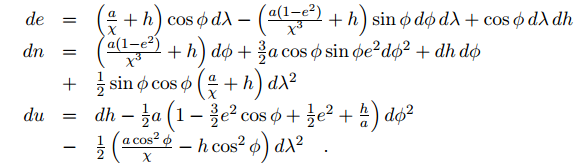
Для преобразования малых изменений используются разложения Тейлора в геоцентрической системе координат:



Координаты  ориентированы, что  сонаправлен с главным меридианом,  направлен под углом  к ввектору  ,  направление на север. Ориентация локальной декартовой системы координат определяется вращением геоцентрической системы координат:



Окончательно получаем выражение для малых изменений координат в локальной декартовой системе координат:



## Алгоритмы, используемые в работе компонента GUI

Взаимодействие между компонентом GUI и библиотекой Radar.lib осуществляется при помощи сигнално-слотовых соединений, реализованных при помощи фреймворка Qt. Установка соединений производится в момент инициализации объектов. Большинство установленных соединений существует до уничтожения объектов интерфейса, то есть до момента завершения выполнения программы. Исключение составляет соединение *SIGNAL(QFileSystemModel::rowsInserted) –SLOT(dataProcessor::readImagesAndRun),* отвечающего за вызов метода обработки радарных изображений при их поступлении в файловую систему. Данное соединение может разрываться в случаях, указанных на диаграмме автоматов, описывающей состояния системы и переходы между ними.

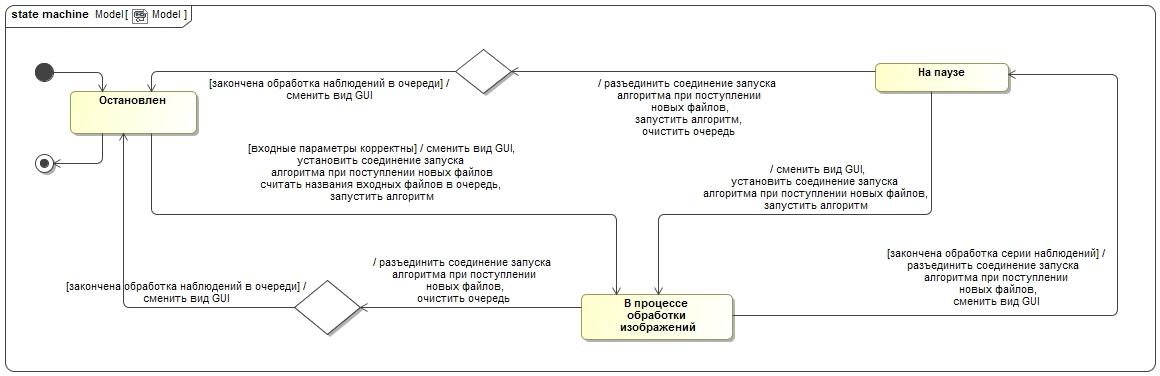


Рисунок 2 – Диаграмма автомата

Состояния системы («Остановлен», «На паузе», «В процессе обработки изображений») изображаются в прямоугольниках. Они отражают абстрактные состояния, в которых находится система в тот или иной момент времени. Указанные состояния интуитивно соответствуют ожиданиям пользователя, взаимодействующего с программой при помощи графического интерфейса. В каждый из этих моментов система полностью определяется своим контекстом.

Надписи на стрелках, соединяющих различные состояния, отражают операции, совершаемые компонентом GUI, необходимые для перехода из одного состояния в другое.

Программа *WhereTheSea* для своей работы три потока, которые обеспечивают:

* Работу интерфейса
* Работу класса *QFileSystemModel*
* Обработку изображений библиотекой *Radar.lib*

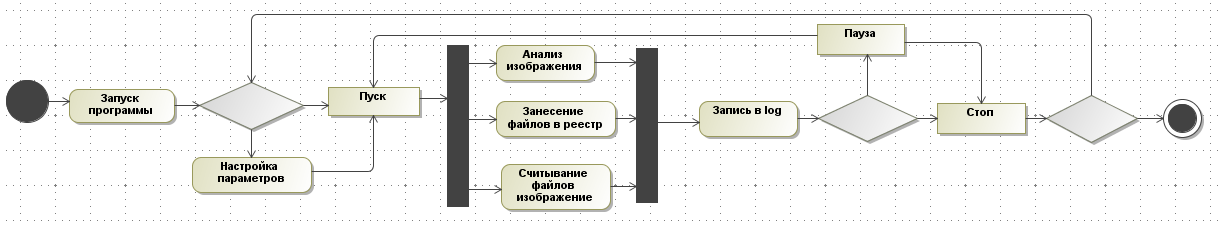


Рисунок 3 – Диаграмма деятельности

На рисунке 3 изображена диаграмма деятельности, отражающая последовательность действий, совершаемую программой.

Выполнение операций с интерфейсом возможно только в том случае, когда завершается обработка изображений, находящихся в очереди. Это обеспечивает надежную работу пользователя с программой, гарантирующую обработку занесенных в очередь изображений даже в случае некорректных действий пользователя во время их обработки.

Пользователь может выполнять лишь определенный набор операций, определяемый состояниям системы. Подобное ограничение повышает надежность системы не умаляя потребностей пользователя.

# Вызов и загрузка

Взаимодействие с программой осуществляется при помощи интерфейса.

При запуске программы появляется главное окно с основными элементами интерфейса.

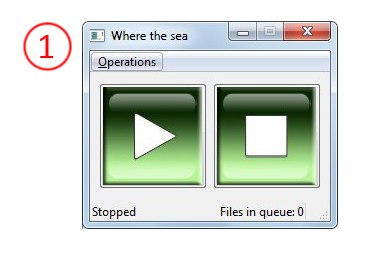


Рисунок 1 - Главное окно

При первом запуске программы необходимо установить соответствующие настройки. Заданные настройки хранятся в регистрах операционной системы при работе в Windows и Linux, что обеспечивает автоматическую установку параметров при последующих запусках программы. Возможность установки параметров доступна при нахождении системы в состоянии «Stopped» и «Paused» (с ограничением на изменение путей)

Для изменения параметров необходимо выполнить команду «Operation» -> «Setting».

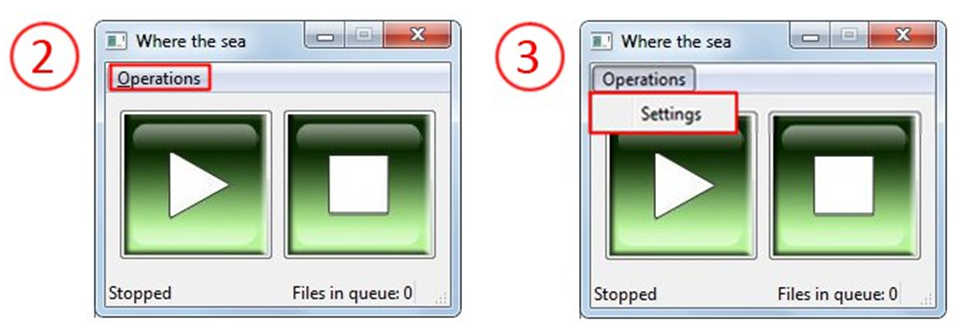


Рисунок 2 – Путь к «Настройкам»

Для корректной работы приложения в появившемся диалоговом окне необходимо указать путь к директории с изображениями для обработки и путь к выходному файлу. При установке флага в поле «Keep a log file» в результате выполнения программы создается лог-файл.

Параметр «Scale» – количество метров в пикселе изображении, зависящий от разрешения входных изображений, формируемых радиолокатором. Принимает значения в диапазоне от 1 до 10000.

Параметр «Identification threshold» – доля шума во входных изображениях. Значение по умолчанию – 0.1. В случае повышенной зашумленности радарных изображений скорость обработки изображений может быть повышена при увеличении значения параметра «Identification threshold», аналогично – при слабом уровне шума параметр может быть уменьшен в целях увеличения производительности. Соответствие между значением, принимаемым параметром, и уровнем шума на изображении указано на рисунке 3. Параметр принимает значения от 0 до 1.

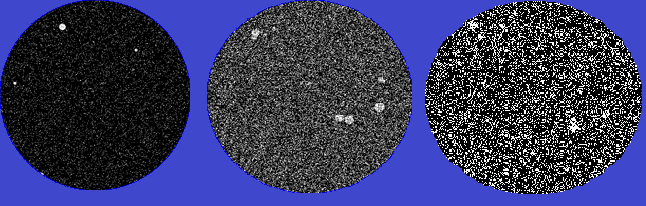


Рисунок 3 – соответствие уровня шума изображений значению параметра «Identification threshold» слева направо: 0.05, 0.15, 0.25.

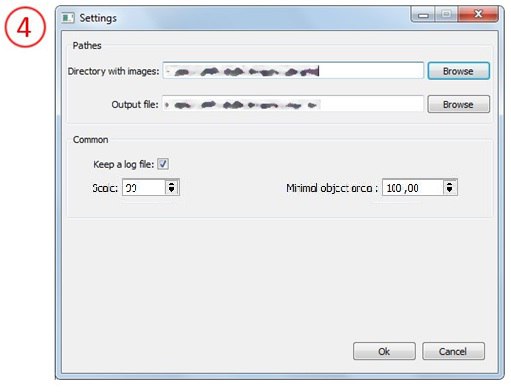


Рисунок 4 - Установка настроек

Для запуска процедуры обработки изображений необходимо нажать кнопку «старт».

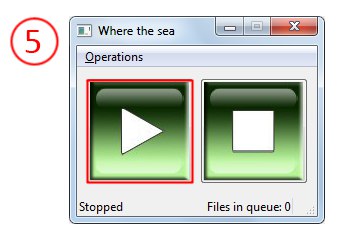


Рисунок 5 - Запуск программы

В процессе работы приложения пользователь может отслеживать количество файлов в очереди на обработку.

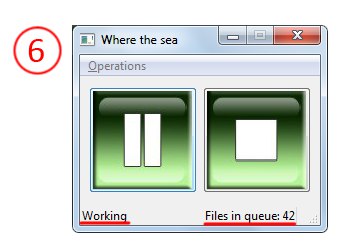


Рисунок 6 - Программа в процессе работы

Для того чтобы приостановить выполнение обработки, необходимо нажать кнопку «пауза». В состоянии «пауза» программа не выполняет обработку изображений, но отслеживает поступление новых файлов с изображениями. При выходе из состояния «паузы» посредством нажатия на кнопку «старт» или «стоп» обрабатываются все изображения, находящиеся в очереди.

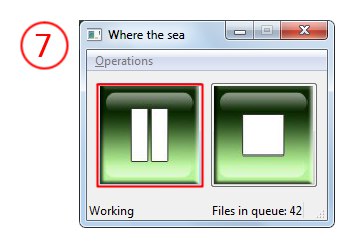


Рисунок 7 - Приостановка выполнения работы

В режиме «пауза» пользователь может видеть, сколько файлов осталось в очереди на обработку, а также он может перейти в настройки и изменить некоторые параметры.

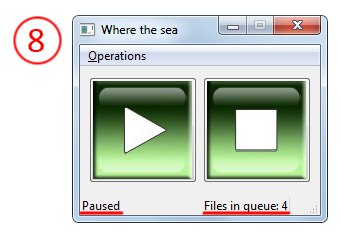


Рисунок 8 - Программа в режиме "пауза"

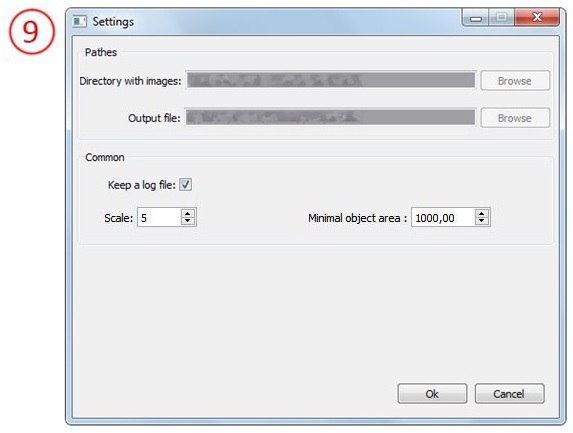


Рисунок 9 - Установка настроек в режиме "пауза"

Для того чтобы остановить работу приложения, необходимо нажать кнопку «стоп». В этом случае прекращается отслеживание новых файлов с изображениями и выполняется обработка всех файлов, находящихся в очереди. По завершению обработки файлов возможно выполнение других операций с программой.

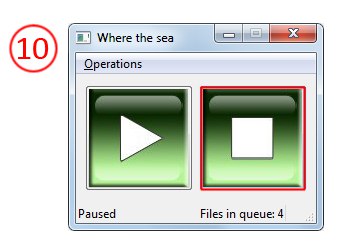


Рисунок 10 - Остановка выполнения работы

В случаях некорректного заполнения полей настроек программа выдает ошибки, указанные на рисунке 11 и 12.

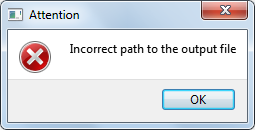


Рисунок 11 - Окно ошибки при некорректном пути к выходному файлу

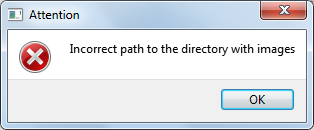


Рисунок 12 - Окно ошибки при некорректном пути к обрабатываемым файлам

# Входные данные

На входе программа получает радарное изображение формата BMP (рисунок 13). Формат имени входного файла имеет вид:

YYYY-MM-DD-hh-mm-ss-dss\_LatSn-LatD-LatM-LatS\_LonSn-LonD-LatM-LatS.bmp*,*

где

|  |  |
| --- | --- |
| [I4] YYYY | год наблюдения |
| [I2] MM | месяц наблюдения |
| [I2] DD | день наблюдения |
| [I2] hh | час наблюдения |
| [I2] mm | минута наблюдения |
| [I2] ss | секунда наблюдения |
| [I] dss | доли секунд наблюдения |
| [1c]LatSn | знак широты центра изображения (- для южного полушария, + для северного полушария) |
| [I3] LatD | градусы широты центра изображения |
| [I2] LatM | минуты широты центра изображения |
| [D] LatS | секунды широты центра изображения |
| [1c]LonSn | знак долготы центра изображения (- для западного полушария, + для восточного полушария) |
| [I3] LonD | градусы долготы центра изображения |
| [I2] LonM | минуты долготы центра изображения |
| [D] LonS | секунды долготы центра изображения |

Таблица – Составляющие формата имени входного файла

Файлы с иным форматом имени не могут быть обработаны корректно. В этом случае производится соответствующая запись в лог-файл.

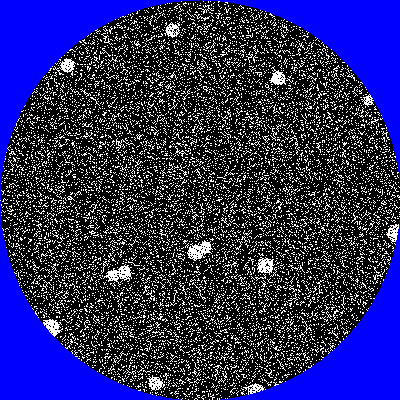


Рисунок 13 – Пример входного файла с изображением

# Выходные данные

Основным файлом, формируемым программой на выходе, является файл, содержащий информацию о характеристиках движения и размере наблюдаемых объектов (рисунок 15). Каждый информационный блок отражает информацию об объектах, наблюдаемых на одном изображении.

|  |
| --- |
| Time  “noise”=noise  Size LatSn LatD LatM LatS LonSn LonD LonM LonS Vx Vy  Size LatSn LatD LatM LatS LonSn LonD LonM LonS Vx Vy  Size LatSn LatD LatM LatS LonSn LonD LonM LonS Vx Vy  …  Size LatSn LatD LatM LatS LonSn LonD LonM LonS Vx Vy |

Рисунок 14 - Формат блока выходного файла

|  |  |
| --- | --- |
| Time = YYYY-MM-DD-hh-mm-ss-dss , где | |
| [I4] YYYY | год наблюдения |
| [I2] MM | месяц наблюдения |
| [I2] DD | день наблюдения |
| [I2] hh | час наблюдения |
| [I2] mm | минута наблюдения |
| [I2] ss | секунда наблюдения |
| [I] dss | доли секунд наблюдения |
| noise - доля шума от общего фона изображения (от 0 до 1 ) | |
| [D15] Size | площадь объекта (метр^2) |
| [1c]LatSn | знак широты (- для южного полушария, + для северного полушария) |
| [I3] LatD | градусы широты объекта |
| [I2] LatM | минуты широты объекта |
| [D15] LatS | секунды широты объекта |
| [1c]LonSn | знак долготы (- для западного полушария, + для восточного полушария) |
| [I3] LonD | градусы долготы объекта |
| [I2] LonM | минуты долготы объекта |
| [D15] LonS | секунды долготы объекта |
| [D15] Vx | скорости объекта по оси OX в локальной системе координат (м/c) |
| [D15] Vy | скорости объекта по оси OY в локальной системе координат (м/c) |

Таблица – Составляющие формата блока выходного файла

|  |
| --- |
| 2017-05-12-12-00-03-00  noize = 0.093546  229679.000000 +50 1 59.853451 1 2 19.569070 -53.300574 -31.183500  107632.000000 +50 1 38.368189 1 3 15.884833 +64.767857 -54.526786  148955.000000 +50 4 48.947437 1 3 31.613110 -46.195833 -87.511111  256587.000000 +50 0 50.465646 1 4 18.551296 -134.043306 +57.636431  69192.000000 +50 6 33.606272 1 7 26.677165 -40.041667 -46.155556  49972.000000 +50 5 2.375915 1 8 5.716492 +119.870280 -89.238811  12493.000000 +50 6 13.254985 1 8 30.238520 UNKNOWN UNKNOWN  2017-05-12-12-00-04-00  noize = 0.093281  237367.000000 +50 1 58.557596 1 2 17.769454 -35.797063 -40.054817  105710.000000 +50 1 36.871275 1 3 18.256852 +47.244805 -46.218182  143189.000000 +50 4 46.293567 1 3 29.430244 -43.340940 -82.029530  246016.000000 +50 0 52.601831 1 4 12.624823 -118.052800 +65.895863  82646.000000 +50 6 31.437642 1 7 22.875815 -75.427326 -67.126615  34596.000000 +50 4 58.908838 1 8 13.673281 +158.378205 -106.844017  37479.000000 +50 6 14.599672 1 8 24.362586 -116.846154 +41.333333 |

Рисунок 14 – outputFile

Скорость объекта может принимать значение UNKNOWN в одном из следующих случаев:

- изображение было первым в серии наблюдений

- объект наблюдается впервые

- объект не удалось сопоставить ни с одним из объектов на предыдущем изображения

|  |
| --- |
| Log file <C:/Users/ Admin /Documents/QtProjects/WhereTheSea/ project/WhereTheSea/gui/ Debug/WhereTheSea\_20170602\_160725.log> creation: 2017-06-02 16:07:25  Directory with images: C:/Users/Admin/Documents/QtProjects/WhereTheSea/inputImages  Output file with objects' motion data: C:/Users/Admin /Documents/QtProjects/WhereTheSea/ project/output.txt  Minimal amount of files to perform: 1  Scale factor: 27  Identification threshold : 0.840000  Set new output file C:/Users/Admin /Documents/QtProjects/WhereTheSea/project/ output.txt  Read images: 2017-06-02 16:07:26  2017-05-12-12-00-00-00\_50-09-12.12124\_01-08-13.9401.bmp  run:  inputFileNames = 2017-05-12-12-00-00-00\_50-09-12.12124\_01-08-13.9401.bmp |

Рисунок 15 – logfile