

Министерство образования Республики Беларусь  
Учреждение образования «Белорусский государственный университет  
информатики и радиоэлектроники»

Факультет телекоммуникаций

Кафедра сетей и устройств телекоммуникаций

Дисциплина: Объектно-ориентированное программирование

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему

**АНАЛИЗ АЛГОРИТМОВ СОВМЕЩЕНИЯ ИЗОБРАЖЕНИЙ,  
ОСНОВАННЫХ НА ВЫДЕЛЕНИИ ГРАНИЦ**

Студент: гр. 663101 Горбуков Андрей Дмитриевич

Руководитель: Макейчик Екатерина Геннадьевна

Минск 2017

## СОДЕРЖАНИЕ

Введение . . . . .	5
1 Описание среды разработки . . . . .	7
1.1 Visual Studio . . . . .	7
1.2 OpenCV . . . . .	8
2 Теоретический обзор используемых алгоритмов . . . . .	11
2.1 Оператор Робертса . . . . .	11
2.2 Оператор Собеля . . . . .	11
2.3 Оператор Превитта . . . . .	12
3 Разработка программы . . . . .	13
3.1 Функциональное проектирование лабораторного стенда . . .	13
4 Анализ полученных результатов . . . . .	16
4.1 Оценка некоторых параметров . . . . .	16
Список использованных источников . . . . .	17

## ВВЕДЕНИЕ

Гиперспектральная съемка развивающиеся и перспективное направление дистанционного зондирования земной поверхности. В основе данного вида съемки лежит понятие гиперспектрального изображения. Гиперспектральное изображение – это трехмерный массив данных, который хранит двухмерную информацию о пространственных координатах и дополнительные одномерные данные о спектральных характеристиках точки поверхности. Данная технология широко применяется в различных сферах, таких как:

- сельское хозяйство;
- медицина;
- производство и переработка продуктов питания;
- минералогия;
- физика;
- астрономия;
- химическая визуализация.

Существует множество различных методов получения данного типа изображения, один из которых использует статический Фурье-спектрометр на основе интерферометра Саньяка. Данный метод налагает свои ограничения на данные получаемые во время съемки. Для построения гиперспектрального изображения требуется накопить достаточное количество данных о точке местности, в процессе ее прохождения вдоль направления полета. Ввиду того, что в дальнейшем над полученными сигналами выполняется дискретное преобразование Фурье, к ним предъявляются строгие требования. В частности необходимо чтобы полученный сигнал относился к одному и тому же участку местности, а его частота дискретизации была равномерной. В лабораторных и других стабильных условиях съемки, когда частоту кадров можно синхронизировать со скоростью полета, достаточно смещать каждый кадр на один пиксель. Таким образом легко получить информацию о яркости любого пикселя под разными углами съемки. При съемке с различных летательных аппаратов вибрации и колебания не позволяют использовать методы применяемые в камерах установленных на спутниках. Для

сведения задачи к вышеописанному случаю требуется отдельно обработать получаемые при съемке местности данные. Для этого необходимо совместить соседние кадры и узнать их относительное перемещение.

Основной задачей данной работы является реализация и анализ различных алгоритмов позволяющих решить вышеописанную проблему.

Предполагается, что использование детекторов границ позволит выделить особые участки изображений остающихся на соседних кадрах и отражающие взаимное смещение между ними. Тем самым сокращая перебор.

# 1 ОПИСАНИЕ СРЕДЫ РАЗРАБОТКИ

В данном разделе будут рассмотрены различные инструменты используемые в ходе выполнения работы, их процесс установки и специфика использования. Также будет произведен краткий сравнительный анализ аналогичных средств, будет дано обоснование выбора именно тех средств, которые использованы в данной работе.

## 1.1 Visual Studio

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

Visual Studio является одним из лидеров среди интегрированных сред разработки на языке C++. Высокая популярность данной среды выражается в огромном мировом сообществе разработчиков использующих её. Благодаря этому у разработчика, применяющего Visual Studio для решения своих задач, появляется возможность более быстрого решения часто возникающих проблем. Также большинство сторонних библиотек и других инструментов имеют версию специально выпущенную для Visual Studio. Большой ассортимент плагинов и расширений доступен в официальном онлайн магазине Microsoft. Почти все они доступны бесплатно. Данные дополнения позволяют упростить разработку и интегрироваться со сторонними сервисами. В частности в ходе разработки данного проекта применялся плагин VisualSVN обеспечивающий возможность работать с системой контроля версий SVN.

Visual Studio является коммерческим проектом, однако для некоммерческой разработки предоставляется специализированная версия

Community. По сравнению с версией Professional данная версия обладает ограниченным функционалом, однако в работе этот функционал не использовался. Поэтому разработка велась на версии Visual Studio 2017 Community RC.

Для начала установки требуется скачать установщик с официального сайта Microsoft<sup>1)</sup>. Там располагается ссылка для скачивания. Запускаем скачанный файл, он запустит процесс установки. Предварительно попросив принять условия лицензионного соглашения. Для перехода на следующий этап требуется нажать кнопку *«Продолжить»*. После этого запустится Visual Studio Installer, который служит для установки и модернизации различных версий Visual Studio 2017.

После выбора нужной версии Visual Studio и нажатия кнопки *«Установить»* откроется окно выбора комплектации среды разработки и дополнительных приложений, показанное на рисунке 1.1. Для работы с данным проектом достаточно выбрать пункт *«Разработка классических приложений на C++»* и Предпочтительные языковые пакеты на вкладке *«Языковые пакеты»*. Однако в зависимости от потребностей пользователя можно установить и дополнительные пакеты. Нажатие на кнопку *«Установить»* запустит процесс установки Visual Studio. После его завершения можно запустить среду разработки. При первом запуске запрашивается вход под аккаунтом Microsoft, если он еще не зарегистрирован то можно или пропустить данный шаг и работать в неавторизированной версии или зарегистрироваться на официальном сайте Microsoft.

## 1.2 OpenCV

*OpenCV* – выпущенная под лицензией BSD библиотека, а следовательно бесплатная для коммерческого и академического использования. Она имеет интерфейсы для C++, C, Python и Java. Поддерживает такие операционные системы как: Windows, Linux, Mac OS, Android и IOS. OpenCV была разработана для обеспечения эффективных вычислений с ориентацией на приложения реального

---

<sup>1)</sup><https://www.visualstudio.com/ru/downloads/>

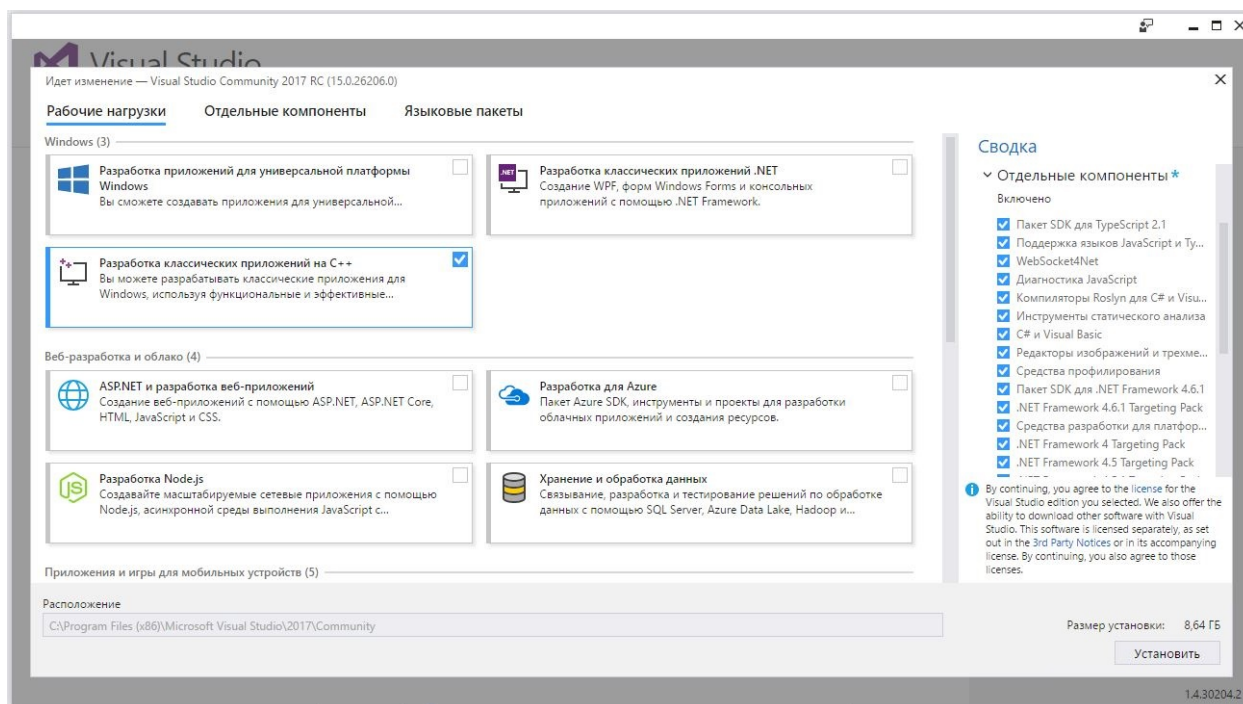


Рисунок 1.1 – Окно выбора комплектации среды разработки и дополнительных приложений

времени. Написанная на оптимизированном C/C++, библиотека способна использовать преимущества многоядерной обработки. Работая вместе с OpenCL она может использовать возможности аппаратного ускорения. Используемая во всем мире, OpenCV насчитывает более 47 тысячи человек в сообществе пользователей и более девяти миллионов скачиваний. Применяется в различных сферах, начиная с интерактивного искусства, обнаружения мин или построения спутниковых карт до современной робототехники [1].

*OpenCV* – наиболее полная, постоянно развивающаяся библиотека для обработки изображений. Огромное количество функций реализующих различные алгоритмы для обработки и анализа изображений позволяют сосредоточиться на решении конкретной задачи, не отвлекаясь на написание заново уже известных классических алгоритмов. Наиболее сильным конкурентом OpenCV в сфере цифровой обработки изображений является MATLAB. Однако он больше подходит для прототипной разработки и исследований алгоритмов из-за удобства отладки, в скорости работы конечного приложения MATLAB сильно уступает возможностям OpenCV. Остальные аналоги реализуют ее функционал лишь частично

и оптимизированы для определенной сферы применения Исходя из этих преимуществ OpenCV выбрана в качестве основного инструмента для разработки данного проекта.



## 2 ТЕОРЕТИЧЕСКИЙ ОБЗОР ИСПОЛЬЗУЕМЫХ АЛГОРИТМОВ

В целях анализа возможности применения детекторов границ для решения задачи совмещения изображения были отобраны три оператора. Основанием для отбора этих операторов послужила их простота реализации и скорость работы. Качество найденных границ не учитывалось, ввиду того, что нет необходимости в точности выделенных объектов, требуется только найти особые области сохраняющие свою форму, ориентацию между кадрами и свое положение относительно земной поверхности.

Все ниже описанные операторы используют различные матрицы для приближенного вычисления первой производной изображения, таким образом резкие границы изменения яркости легко обнаружить на основании того, что их первые производные превосходят по модулю некоторый порог.

### 2.1 Оператор Робертса

Один из старейших детекторов разработанный Л. Робертсом в 1963 году [2]. По своей сути вычисляет сумму квадратов разниц между диагонально смежными пикселями. Это может быть выполнено сверткой изображения с двумя ядрами:

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix} \quad (2.1)$$

### 2.2 Оператор Собеля

Дискретный дифференциальный оператор, вычисляющий приближённое значение градиента яркости изображения. Результатом применения оператора Собеля в каждой точке изображения является либо вектор градиента яркости в этой точке, либо его норма [3].

Оператор выполняет свертку двумя ядрами  $3 \times 3$  для вычисления приближённых значений производных по горизонтали и по вертикали.

Пусть  $\mathbf{A}$  – исходное изображение, а  $\mathbf{G}_x$  и  $\mathbf{G}_y$  приближенные производные вычисляемые следующим образом:

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \mathbf{A}; \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \mathbf{A} \quad (2.2)$$

Где  $*$  обозначает двумерную операцию свертки.

Приближенное значение модуля градиента можно вычислить путем взятия квадратного корня из суммы квадратов соответствующих элементов ранее вычисленных матриц.

$$G = \sqrt{G_x^2 + G_y^2} \quad (2.3)$$

Дополнительно оператор Собеля позволяет вычислить направление градиента:

$$\theta = \arctan \left( \frac{G_y}{G_x} \right) \quad (2.4)$$

### 2.3 Оператор Превитта

Оператор Превитта использует следующие маски для приближения частных производных:

$$\mathbf{G}_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} * \mathbf{A}; \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} * \mathbf{A} \quad (2.5)$$

Данные маски аналогичны оператору Собеля и отличаются только коэффициентом 2 при средних элементах. Данные изменения влияют на уровень сглаживания.

### 3 РАЗРАБОТКА ПРОГРАММЫ

#### 3.1 Функциональное проектирование лабораторного стенда

Программа реализована с использованием методологий ООП, что позволило повысить коэффициент повторного использования кода и улучшить его читаемость. В результате разработанные классы могут быть использованы при дальнейших разработках программ схожей тематики. Как можно увидеть из рисунка 3.1 программа состоит из пяти классов, один из которых абстрактный и один статический.

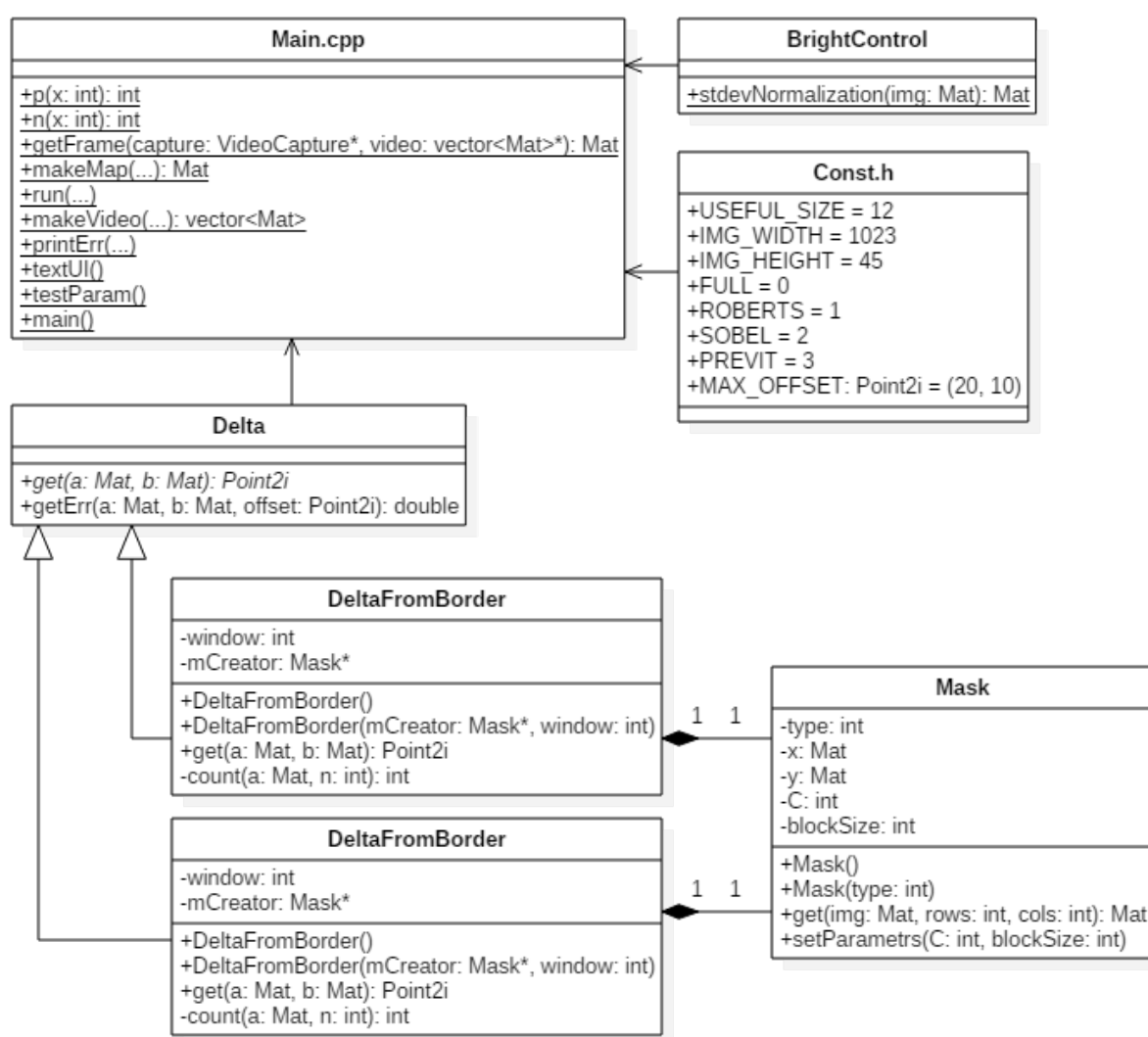


Рисунок 3.1 – Структура программы

##### 3.1.1 Main.cpp

Содержит функции , реализующие основную логику программы и чтение/запись файлов. В некоторых из данных функций встречаются одинаковые параметры:

*capture* – входной видеопоток из которого происходит чтение данных.

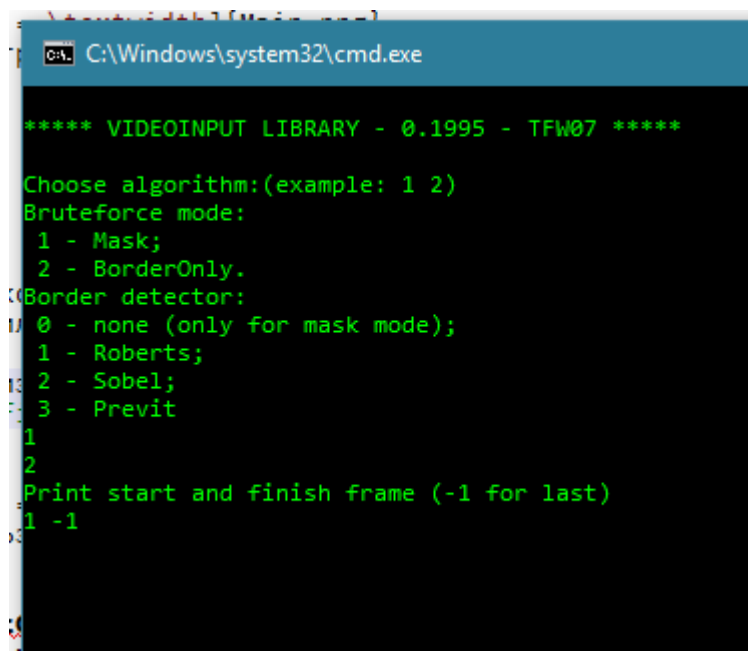
*frameSize* – размер кадра оригинального видео.

*video* – массив изображений в который покадрово записывается видео готовое к обработке.

*offsets* – массив относительных смещений кадров, вычисляемых в процессе работы алгоритмов.

*main()* – точка входа в программу. В зависимости от варианта сборки запускает функцию *textUI()* или *testParam()*.

*textUI()* – реализует текстовый интерфейс для взаимодействия с пользователем. Как видно на рисунке 3.2 пользователь может выбрать тип маски и алгоритм, а также диапазон обрабатываемых кадров.



```
***** VIDEOINPUT LIBRARY - 0.1995 - TFW07 *****

Choose algorithm:(example: 1 2)
Bruteforce mode:
  1 - Mask;
  2 - BorderOnly.
Border detector:
  0 - none (only for mask mode);
  1 - Roberts;
  2 - Sobel;
  3 - Previt
1
2
Print start and finish frame (-1 for last)
1 -1
```

Рисунок 3.2 – текстовый пользовательский интерфейс

*testParam()* – служит для многократного запуска алгоритма. Использование данной функции позволяет удобно тестировать алгоритмы при различных параметрах и собирать статистические данные.

*vector<Mat> makeVideo( VideoCapture \*capture, Size2i \*frameSize, int start = 1, int length = -1)* – загружает видео в виде вектора изображений из

видео потока *capture*. Также возвращает размеры кадра. Может принимать номер первого кадра и длину получаемого видео.

*Mat getFrame( VideoCapture \*capture, vector<Mat> \*video)* – читает следующий кадр из видеопотока, подготавливает его и записывает в массив кадров.

*run( Size2i frameSize, vector<Mat> \*video, Mat \*map, Delta \*delta, vector<Point2i> \*offsets)* – обрабатывает видео и запускает создание результирующего изображения *map*. Для этого принимает объект *delta* определяющий каким алгоритмом будет вычислено относительное смещение кадров.

*Mat makeMap( Size2i frameSize, vector<Mat> \*video, vector<Point2i> \*offsets, int maxL, int maxR, int maxD, int maxU)* – создает результирующее изображение, для этого требуется узнать максимальные абсолютные смещения кадров во всех четырех направлениях.

*int printErr( vector<Mat> \*video, vector<Point2i> \*offsets, ofstream \*sout, Mask \*mCreator2)* – выводит в текстовый файл дополнительные статистические данные.

### **3.1.2 Класс BrightControl**

### **3.1.3 Абстрактный класс Delta**

### **3.1.4 Класс DeltaBruteForce**

### **3.1.5 Класс DeltaFromBorder**

### **3.1.6 Класс Mask**

### **3.1.7 Const.h**

## **4 АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ**

### **4.1 Оценка некоторых параметров**

#### **4.1.1 Нахождение части кадра пригодной для обработки**

#### **4.1.2 Максимальная амплитуда колебаний камеры**

### **4.1.3 Параметры порогового фильтра**

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Itseez. OpenCV [Электронный ресурс]. — Режим доступа : <http://opencv.org/>.

[2] Roberts, Lawrence G. Machine perception of three-dimensional solids: phdthesis. — 1963. — 82 P.

[3] Sobel, Irwin. History and Definition of the so-called "Sobel Operator more appropriately named the Sobel-Feldman Operator [Электронный ресурс]. — Режим доступа : <https://www.researchgate.net/publication/>.