**УТВЕРЖДАЮ**

Должность

\_\_\_\_\_\_\_\_\_\_\_\_\_ ФИО

“ ” \_\_\_\_\_\_\_\_\_ 2018 г.

**Пояснительная записка № 004**

**Этап 4. Программная реализация и тестирование**

**НИР**

**Реконструкция 3D модели поверхности микроскопического объекта по серии изображений**

**(«Get3DModel»)**

**Н.Новгород**

**2018**Реферат

Пояснительная записка 004,страниц 27.

КЛЮЧЕВЫЕ СЛОВА: математическая модель, структура решения, ограничения на решение, эталон, оценка решения, алгоритм решения, схема оценки работы алгоритма, Get3DModel,тестовый базис, эталон.

В пояснительной записке в рамках проекта НИР «Get3DModel» представлены:

* Программная реализация расчетного блока кода;
* Тестовая инфраструктура;
* Тестирование кода;
* тривиальные примеры;
* комбинированные (сложные) примеры;
* эталоны для тривиальных примеров и метрики для сложных примеров;
* код нарезки фрагментов исходных данных (картинок);
* файлы формата .camera;

А также параметры оценки выходных данных:

* суммарная ошибка;
* максимальная ошибка;
* средняя ошибка;
* глубина уровня;
* заполняемость;
* равномерность распределения точек;
* время выполнения;

Оглавление

[Термины и определения 4](#_Toc513987798)

[1 Программная реализация расчетного блока кода 5](#_Toc513987799)

[2 Тестовая инфраструктура 7](#_Toc513987800)

[3 Тестирование кода 20](#_Toc513987801)

[4 Список используемых источников 27](#_Toc513987802)

# Термины и определения

*Get3DModel* – разрабатываемый в рамках текущей НИР ([1]) программный модуль реконструкции 3D модели поверхности микроскопического объекта по серии изображений

*3D изображение* – изображение, полученное путем моделирования объемных объектов в трехмерном пространстве.

*Градиент* -  вектор, своим направлением указывающий направление наибольшего возрастания некоторой величины, значение которой меняется от одной точки пространства к другой (скалярного поля), а по величине (модулю) равный скорости роста этой величины в этом направлении.

*Тривиальные примеры* – примеры входных данных; набор изображений одинакового размера (формат.png размером не больше 1К), полученный микросъемкой одного и того же объекта с разной высоты.

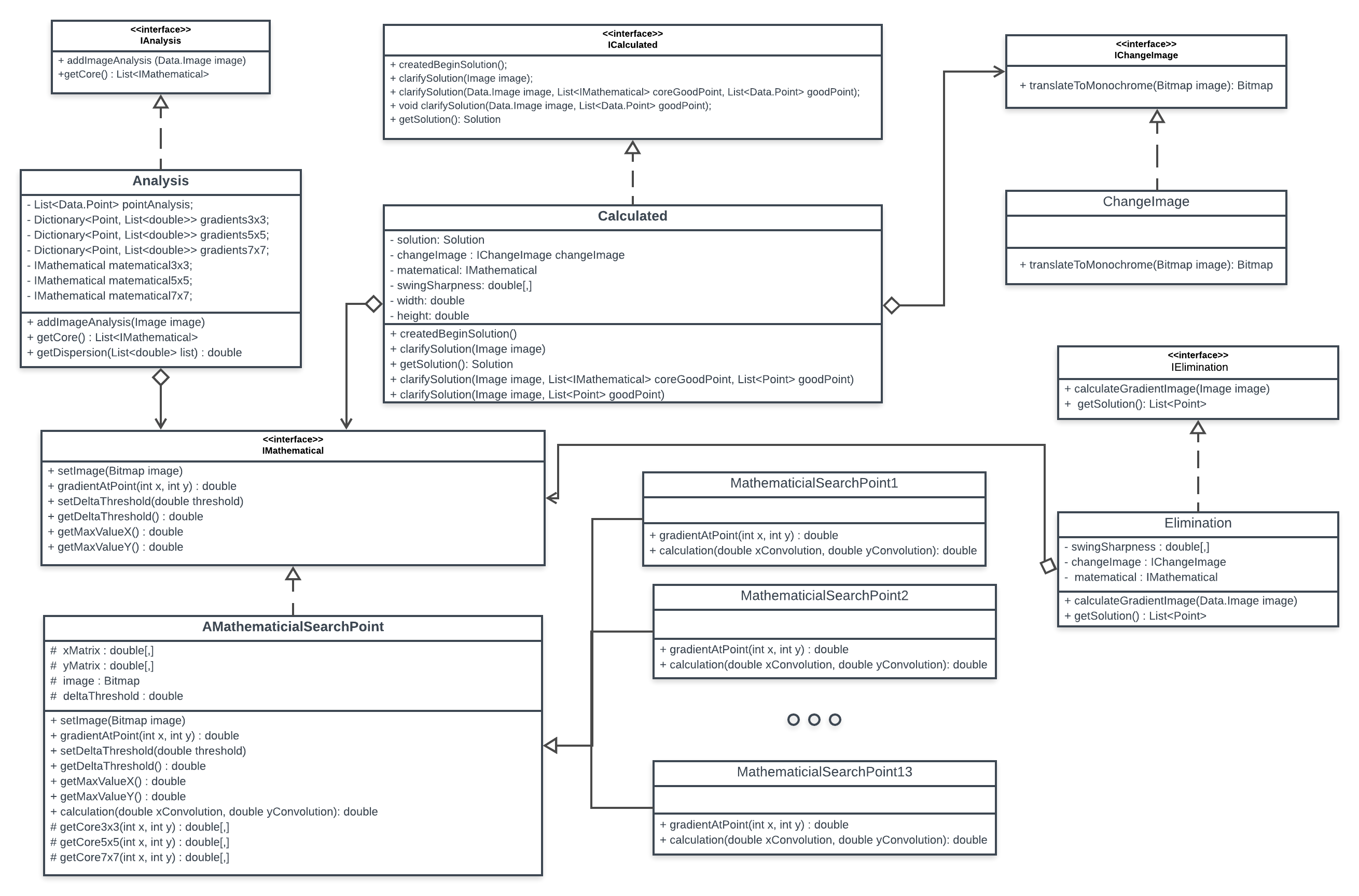
*Комбинированные (сложные) примеры -* примеры входных данных; набор изображений одинакового размера (формат.png размером не больше 4К), полученный микросъемкой одного и того же объекта с разной высоты, содержащие в себе комбинации сложных для анализа фрагментов (блики, размытость, затемнения и тд.);

*Ошибка* – модуль разности высот соответствующих точек из эталонной и полученной моделей.

# Программная реализация расчетного блока кода

Рассматриваемый блок состоит из 5 логических составляющих:

1. Класс для динамического подбора ядер.  
   Интерфейс: IAnalysis  
   Класс реализующий интерфейс: Analysis  
   Основной функционал:
   1. Добавить изображение для анализа  
      voidaddImageAnalysis (Data.Imageimage);  
      image : изображение анализа
   2. Получить ядра для каждой точки  
      List<IMathematical>getCore();
2. Класс для обработки изображения.  
   Интерфейс: IChangeImage  
   Класс реализующий интерфейс: ChangeImage  
   Основной функционал:
   1. Перевод изображения в монохром  
      BitmaptranslateToMonochrome(Bitmapimage);  
      image : изображение для перевода
3. Класс для отсева точек не несущих достоверную информацию  
   Интерфейс: IElimination  
   Класс реализующий интерфейс: Elimination  
   Основной функционал:
   1. Рассчитать градиенты на изображении  
      voidcalculateGradientImage(Imageimage);  
      image : изображение для расчёта
   2. Получить достоверные точки  
      List<Data.Point>getSolution();
4. Класс для математических подсчетов  
   Интерфейс: IMathematical  
   Абстрактный класс реализующий интерфейс: MathematicialSearchPoint  
   Список классов наследуемых от абстрактного, реализующие разные варианты ядер для подсчета градиента: MathematicialSearchPoint1 - MathematicialSearchPoint13  
   Основной функционал:
   1. Установка изображения для расчетов  
      voidsetImage(Bitmapimage);  
      image : изображение для расчёта
   2. Подсчет градиента в точке  
      doublegradientAtPoint(intx, inty);  
      [x,y] : координаты точки
   3. Установить пороговое значение  
      voidsetDeltaThreshold(doublethreshold);  
      threshold : пороговое значение
   4. Получить порогое значение  
      doublegetDeltaThreshold();
   5. Установить пороговое значение  
      voidsetDeltaThreshold(doublethreshold);  
      threshold : пороговое значение
5. Основной класс для подсчета решения задачи  
   Интерфейс: ICalculated  
   Класс реализующий интерфейс: Calculated  
   Основной функционал:
   1. Создать начальное решение  
      voidcreatedBeginSolution();
   2. Уточнить решение  
      voidclarifySolution(Imageimage);  
      image: изображение для уточнения решения
   3. Уточнить решение  
      voidclarifySolution(Data.Imageimage, List<IMathematical>coreGoodPoint, List<Data.Point>goodPoint);  
      image: изображение для уточнения решения  
      coreGoodPoint: список ядер для каждой достоверной точки  
      goodPoint: список достоверных точек
   4. Уточнить решение  
      voidclarifySolution(Data.Imageimage, List<Data.Point>goodPoint);  
      image: изображение для уточнения решения  
      goodPoint: список достоверных точек
   5. Получитьрешение  
      Solution getSolution();



# Тестовая инфраструктура

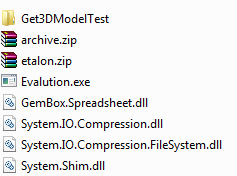
Для анализа эффективности того или иного алгоритма, необходимо решение следующих задач:

1. Программная реализация тестовой инфраструктуры для оценки алгоритмов;
2. Подготовка входных примеров разных размеров для тестирования алгоритмов;
3. Сравнение полученных выходных данных с эталонами (для тривиальных примеров) по параметрам, обговорённым с Заказчиком;
4. Оформление данных, полученных с помощью тестов, в виде таблицы(образец таблицы представлен ниже – таблица 1).

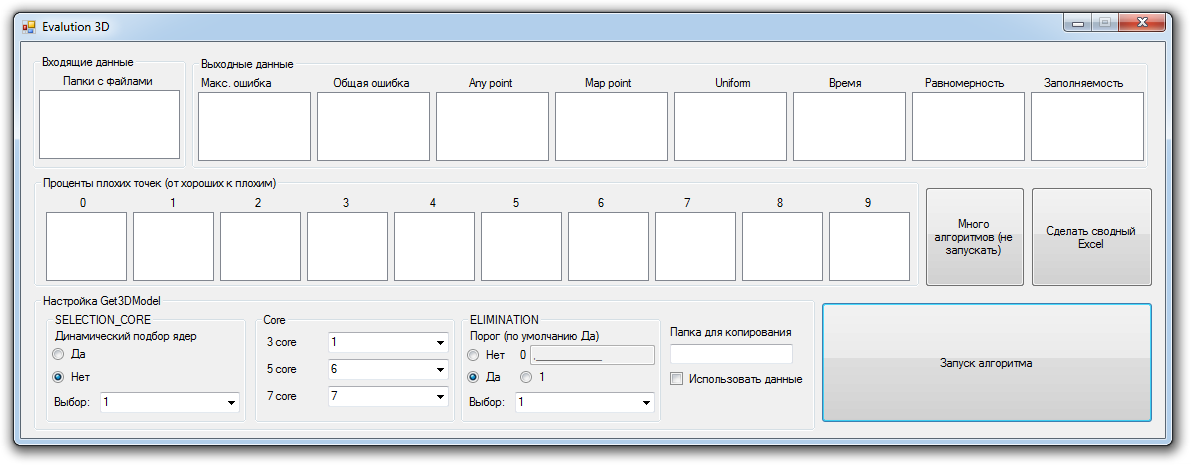
***Описание программной реализации тестовой инфраструктуры***

1. Для корректной работы программы необходимо:

* Программа Evalution.exe;
* 4 DLL-файла: GemBox.Spreadsheet.dll, System.IO.Compression.dll, System.IO.Compression.FileSystem.dllиSystem.Shim.dll;
* Zip-архив archive.zip, в котором находятся папки с картинками и файлом sample.camera в каждой из них;
* Zip-архив etalon.zip, в котором находятся папки с теми же названиями, что и в предыдущем файле, а в этих папках файл etalon.dat – эталонный dat-файл;
* Папка с программой Get3DModelTest, в которой находится соответствующая программа со всем содержимым.



1. Запуск программы:



1. В нижней части программы в группе Настройка Get3DModel устанавливаем настройки в соответствии с файлом «*Настройка для тестового подпроекта с помощью файла настроек setting.ini.docx*».  
   Так же, как и в данном файле, программа настройки программы разделены на 3 части: SelectionCore (динамический подбор ядер), Core (настройка ядер для процедуры отсева) и Elimination (настройка отсева).

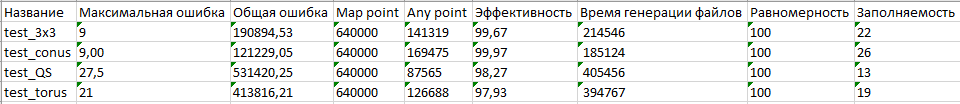
* В блоке SelectionCore есть 2 кнопки radiobutton, отвечающие за выбор использовать или нет динамический подбор ядер. В случае нединамического подбора, существует возможность выбора одной из 11 стратегий (MathematicialSearchPoint1-…- MathematicialSearchPoint11).
* Блок Core имеет 3 настройки, с выбором номеров стратегий для 3х, 5ти и 7ми ядер в соответствии с ТЗ.
* Блок Elimination имеет 3 radiobutton, с помощью которых можно выбрать порог по умолчанию (Да), любой свой в диапазоне от 0<=x<1 (Нет) и порог, равный единице (1). Кроме того, есть выбор стратегии для подсчёта градиента (также номера от 1 до 11).

1. Необходимо также указать название папки для копирования (см. далее).
2. После запуска алгоритма (кнопка «Запуск алгоритма») последовательно будут проделаны следующие действия:

* Создастся папка из текстового поля;
* В неё будет разархивировано содержимое архива *archive.zip;*
* В неё же скопирован файл настроек *setting.ini*, который будет сгенерирован программой;
* Запущена программа Get3DModelTest из папки с программой.
* После окончания работы той программы (около 5-10 минут) будет создана папка «*Results\_НазваниеПапкиДляКопирования*» и разархивирован файл *etalon.zip*
* В программе будут заполнены все оставшиеся поля (группы «Выходные данные» и «Проценты плохих точек (от хороших к плохим)» (см скриншот из пункта 2).

1. Также в папке «*Results\_НазваниеПапкиДляКопирования*» будут созданы Excel-файлы: один файл будет называться «Данные», а остальные будут названы также, как и названия папок.
2. В файле «Данные.xls» есть несколько листов:

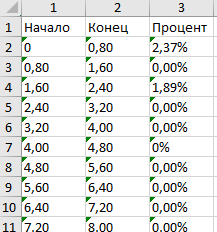
* Лист «*Данные*» содержит сравнительную информацию по всем экспериментам из верхней части программы (Макс ошибка, Общая ошибка, Mappoint, Anypoint, Эффективность, Время генерации, Равномерность, Заполняемость).



* Лист «*Сравнение*» содержит сравнительную информацию о процентах плохих точек



* Остальные листы названы также, как и папки. Там находятся те же данные, что и в предыдущем листе, только в более привычном виде (транспонированном).

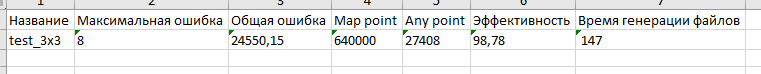


1. Также в той же папке создадутся файлы с названием папок. В каждом из этих файлов будут:

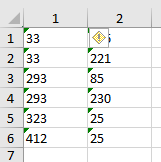
* Лист «*Данные*» это данные по процентам точек (как в пункте 7В)



* Лист «*Статистика*» это одна строка из файла Данные, листа Данные (пункт 7А)



* Листы «*0*»… «*9*» это листы с координатами точек. «0» - это лист с наилучшим совпадением точек, а «9», соответственно, с наихудшим. Первая координата это X, вторая - Y



1. В программе есть ещё способ использовать ранее полученные данные для получения Excel-файлов. Для этого перед запуском алгоритма необходимо поставить галочку checkbox в поле «Использовать данные» и в текстовое поле «Папка для копирования» написать нужное название папки (например, для папки «*Resulst\_0000*» необходимо написать «*0000*»).
2. Для получения сводной таблицы по всем экспериментам можно воспользоваться кнопкой «Сделать сводный Excel». После нажатия этой кнопки будет создан файл "*Сводная.xls*" в папке с программой.
3. Кнопка «Много алгоритмов (не запускать)» работает пока в тестовом режиме, запуск её нежелателен. Она должна простым перебором создать все возможные *setting.ini*–файлы и запустить алгоритмы со всеми этими настройками.

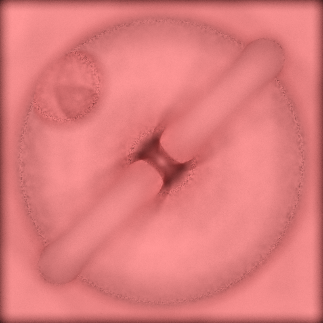
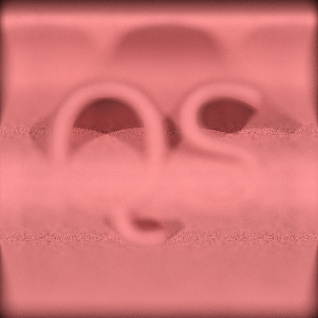
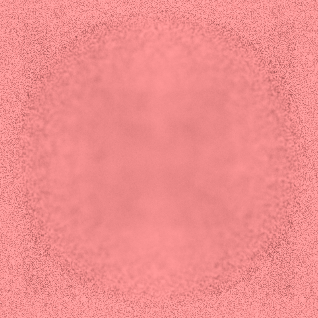
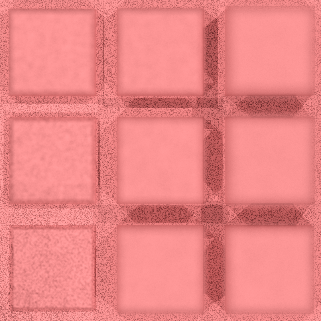
## Тестирование проводилось на разных типах картинок, предоставленных заказчиком:

# *Тривиальные примеры*

Картинки формата .png размером не более 1К.

*Образцы:*

test3x3 test\_conustest\_QStest\_torus



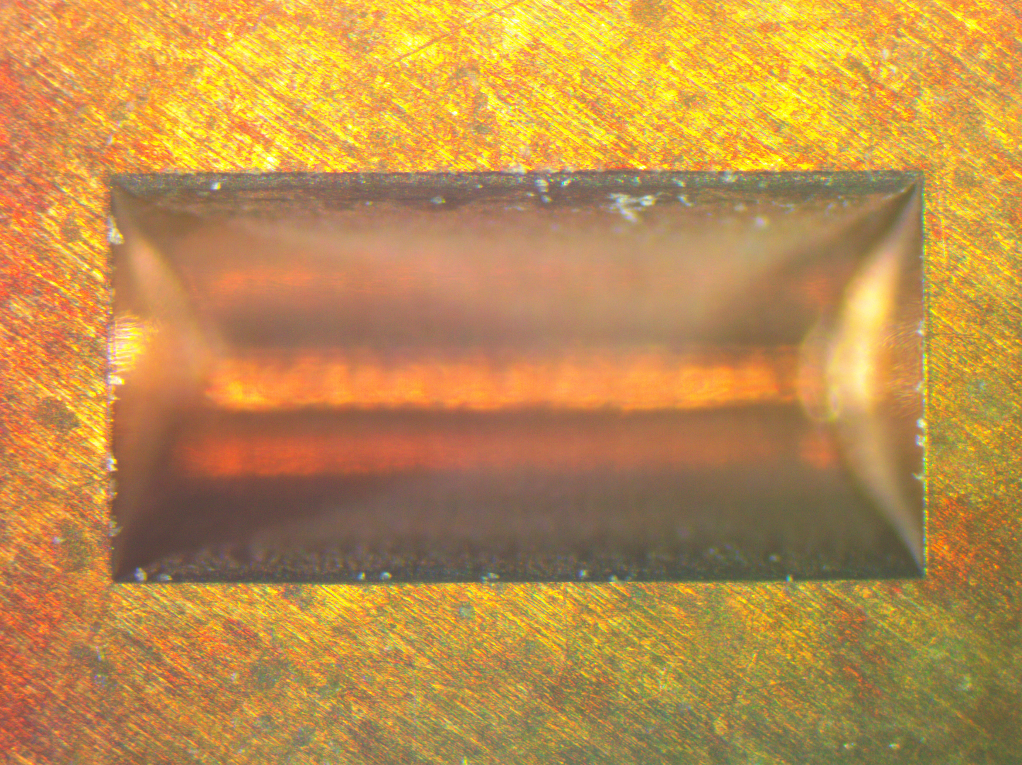
*Эталоны:*

Файлы формата .dat, содержащие матрицу размера, соответствующего входной картинке, где номера строк/столбцов матрицы – координаты точек, ячейки матрицы – высоты соответствующих точек.

# *Комбинированные (сложные) примеры*

Картинки формата .png размером не более 4К

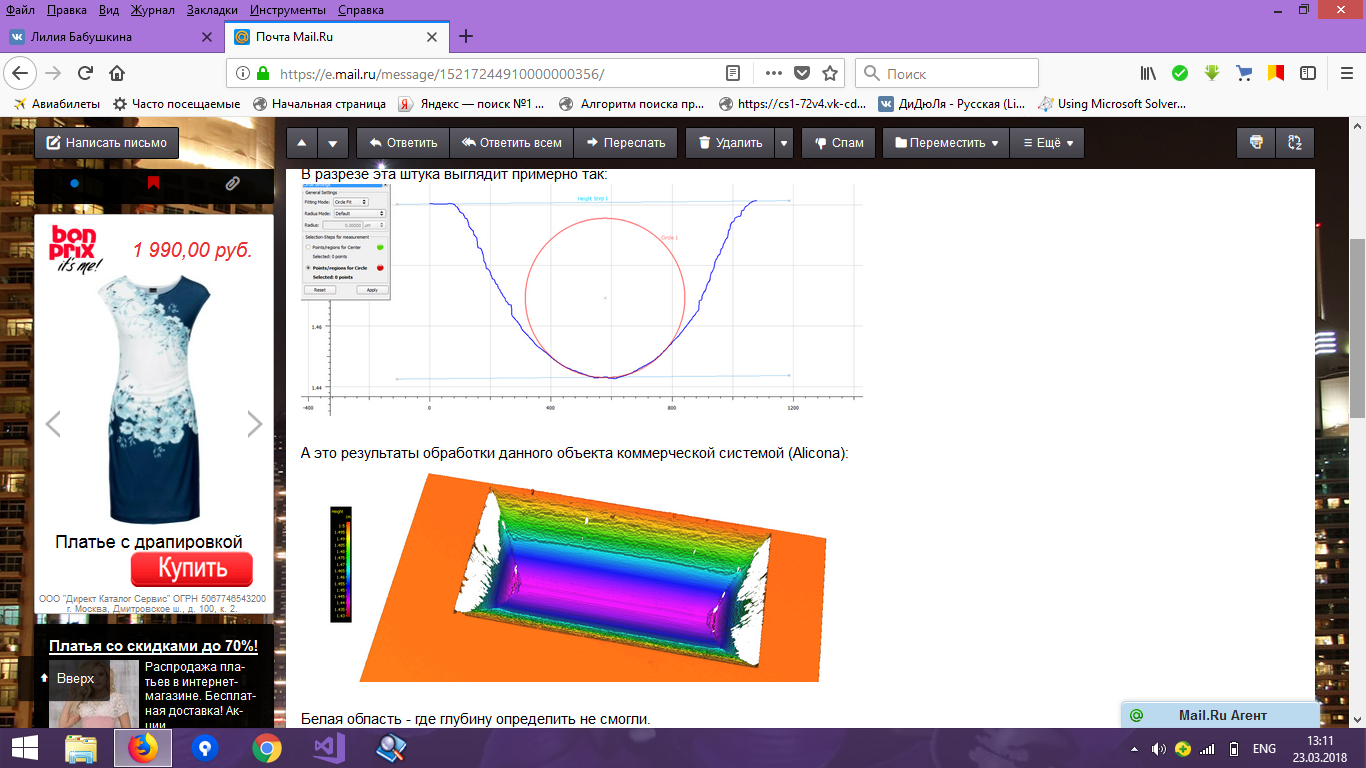
*Образец:* trench\_png



*Эталон:*

В данном примере требований к точности нет, однако, результаты оценки выходной модели по этому тесту должны быть отражены в отчёте в отдельной таблице.

**Метрики, предоставленные Заказчиком**



# *Файлы формата .camera*

параметры оптической системы, при помощи которой были получены изображения поверхности микроскопического объекта (фокусное расстояние, наблюдаемая ширина в фокусе, коэффициент для вычисления абсолютной высоты фокуса);

***Схема оценки выходных данных***

При условии, что ограничения по времени и по заполнению точек выполняются, оценка производится по следующим параметрам (по убыванию значимости для Заказчика):

* **Суммарная ошибка** (Ошибка – модуль разности высот соответствующих точек из эталонной и полученной моделей);

*Математическое описание параметра:*

Пусть и – матрицы t – ого тестового набора, содержащие высоты точек эталонной и полученной моделей соответственно, тогда и – элементы эталонной и полученной матриц соответственно, где r и m – высоты точек с координатами i, j. (r,mϵZ; (i, j) ϵ {}, где – множество точек t-ого тестового набора, для которых (высота точки положительна), t, где p – количество тестовых наборов, x – размеры картинки t – ого тестового набора).

*Индивидуальная оценка теста:*

*Средняя оценка тестового набора:*

* **Максимальная ошибка**

*Индивидуальная оценка теста:*

*Нормированная оценка:*

,

где s=, n=, знаменатель дроби - разность между максимальной и минимальной высотами эталона соответствующего t - ого теста.

*Максимальная ошибка тестового набора:*

,

где s=, n=, и p – количество тестовых наборов.

* **Средняя ошибка**

*Описание параметра:*

Равна отношению суммарной ошибки теста к количеству точек, с высотой

*Индивидуальная оценка теста:*

где – множество точек t-ого тестового набора, для которых (высота точки положительна), и – элементы эталонной и полученной матриц соответственно, где r и m – высоты точек с координатами i, j.

*Нормированная оценка:*

p – количество тестовых наборов.

* **Глубина уровня**

*Описание параметра:*

Равна отношению средней ошибки теста к количеству его (теста) картинок.

*Индивидуальная оценка теста:*

где числитель дроби – средняя ошибка, – количество картинок t – ого, t теста.

*Нормированная глубина уровня:*

p – количество тестовых наборов.

* **Заполняемость**

*Математическое описание параметра:*

Пусть D – множество точек полученной модели, для которых соответствующие высоты больше нуля (>0), D ϵ {SxN}.

*Тогда заполняемость вычисляется по формуле:*

* **Равномерность распределения точек**

*Математическое описание параметра:*

Точки восстанавливаемых объектов, высоты которых найдены алгоритмом, должны быть равномерно распределены по исследуемой области. Для этого необходимо вычислить вектор:

, где

– количество уровней равномерного распределения;

t – количество тестовых наборов;

Координата вектора:

, , где

– количество областей i-ого уровня, содержащих хотя бы одну точку с найденной высотой.

– количество областей i-ого уровня, на которые делим изображение.

Поэтому исходя из параметра равномерности необходимо выполнение следующего условия:

*Индивидуальная оценка теста:*

*Среднее распределение для всех тестовых наборов высчитывается по формуле:*

* **Время выполнения**

Высчитывается время выполнения одного теста. Для оценки алгоритма необходима общая оценка времени выполнения всех тестов. В связи с этим, применяется нормировка времени.

*Математическое описание параметра:*

Пусть – время выполнения i-го теста (i =, где p – количество тестовых наборов);

– число картинок i- го теста;

– количество точек картинки i – го теста.

*Индивидуальная оценка теста:*

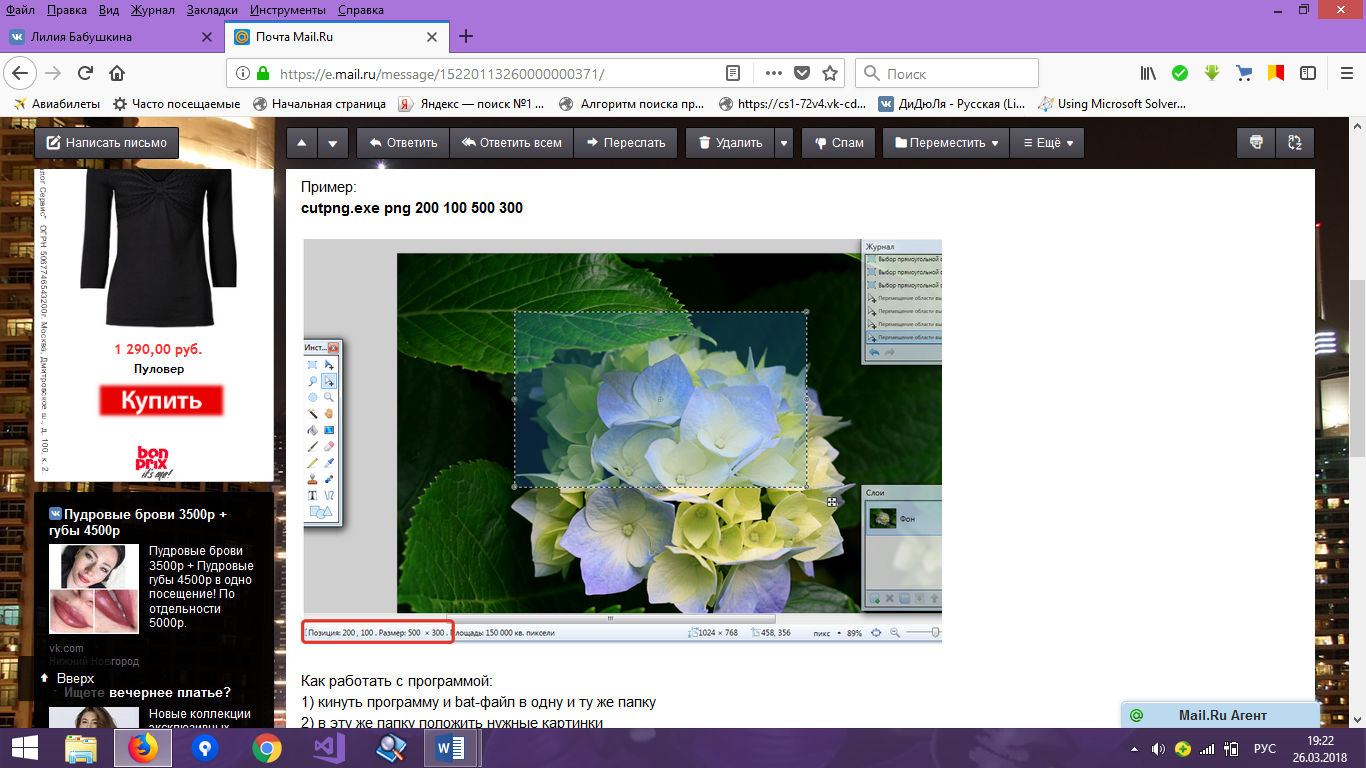
*Средняя оценка тестового набора:*

Результаты будут представлены в виде итоговой таблицы, в которой будут отражены сведения по каждому тесту отдельно, а также нормированные сведения по всем тестам.

***Нарезка фрагментов***

Запуск программы происходит в файле формата .bat, файл содержит:

* название программы;
* формат файлов;
* координаты левого верхнего угла (сначала горизонтальная, затем вертикальная);
* размер вырезаемого изображения;

Пример:**cutpng.exe png 200 100 500 300**

***Оформление результатов тестирования***

*Таблица 1.*

Core 1.10.11

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Название | максимальная ошибка | Суммарная ошибка | Общее число точек | Число точек с положительной высотой | Время генерации файлов | Равномерность | Заполняемость | Средняя ошибка | Глубина уровня | Время нормированное |
| test\_3x3 | 6 | 10301,24 | 640000 | 14441 | 87878 | 100 | 2,25 | 0,71 | 0,07 | 0,01373 |
| test\_conus | 3 | 765,76 | 640000 | 9517 | 85706 | 83,33 | 1,48 | 0,08 | 0,00 | 0,01339 |
| test\_QS | 27,5 | 29611,90 | 640000 | 5009 | 198338 | 66,66 | 0,78 | 5,91 | 0,26 | 0,01408 |
| test\_torus | 14,35 | 5407,93 | 640000 | 8482 | 197547 | 83,33 | 1,32 | 0,63 | 0,02 | 0,01403 |
| ∑ | 12,71 | 11521,70 | 640000 | 9362,25 | 142367,25 | 83,33 | 1,45 | 1,83 | 0,08 | 0,01380 |

# Тестирование кода

Задачей являлось:

1. Создание тестового набора для каждой нетривиальной функции или метода консольного приложения Get3DModel.exe.
2. Прогон программы на тестах с получением отчета по результатам тестов.
3. Оценка результаты выполнения программы на наборе тестов с целью принятия решения о продолжении или остановке тестирования.
4. После очередного изменения кода - регрессионное тестирование (обнаружение ошибок в уже протестированных участках исходного кода).

Целью являлось:

1. Выявление локализованных ошибок в реализации алгоритмов.
2. Выявление ошибок кодирования (ошибки работы с условиями, счетчиками циклов, использования локальных переменных и ресурсов)

Обзор:

1. Расположение тестов:

Все тесты находятся в проекте Get3DModel.UnitTests

1. Тестовые классы.

Именование: [Имя класса]Tests

Каждый тестовый класс тестирует только одну сущность. Класс проекта соответствует одному тестовому классу.

1. Тестовые методы:

Именование: [Название тестируемого методаTest].

1. Оформление теста: «arrange-act-assert» (организация – действие – утверждение).

В модульном тесте чётко определены предусловия (инициализация тестовых данных, предварительные установки), действия (то, что тестируется) и постусловия (что должно быть в результате выполнения действия).

1. Результаты тестирования представлены в таблице:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Тестируемый блок** | **Тестируемый класс** | **Приоритет** | **Тестируемый метод** | **Название тестового метода/Тест-кейс** | **Результат (True\False)** |
| CalculatedBlock | EliminationTests | 1 | void calculateGradientImage(Data.Image image) | calculateGradientImageTest  Проверка подсчета градиентов картинки | True |
| List<Data.Point>getSolution() | getSolutionTest  Проверка получения списка хороших точек (отсева) | True |
| MathematicialSearchPoint1  (2 ядра свертки: 3х3 вычисление градиента: по среднему арифметическому) | 1 | MathematicialSearchPoint1() | MathematicialSearchPoint\_1Test  Проверкаядерсвертки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint1\_CornerPoint\_Test  Проверка градиента угловой точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint1\_BoundaryPoint\_Test  Проверка градиента граничной точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint1\_IntPoint\_Test  Проверка градиента внутренней точки | True |
| MathematicialSearchPoint2  (2 ядра свертки: 3х3 вычисление градиента: по сумме квадратов) | 1 | MathematicialSearchPoint2() | MathematicialSearchPoint\_2Test  Проверка ядер свертки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint2\_CornerPoint\_Test  Проверка градиента угловой точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint2\_BoundaryPoint\_Test  Проверка градиента граничной точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint2\_IntPoint\_Test  Проверка градиента внутренней точки | True |
| MathematicialSearchPoint3  (1 ядро свертки: 3х3 вычисление градиента: по ядру свертки) | 1 | MathematicialSearchPoint3() | MathematicialSearchPoint\_3Test  Проверка ядра свертки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint3\_CornerPoint\_Test  Проверка градиента угловой точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint3\_BoundaryPoint\_Test  Проверка градиента граничной точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint3\_IntPoint\_Test  Проверка градиента внутренней точки | True |
| MathematicialSearchPoint4  (1 ядро свертки: 3х3 вычисление градиента: по ядру свертки) | 1 | MathematicialSearchPoint4() | MathematicialSearchPoint\_4Test  Проверка ядра свертки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint4\_CornerPoint\_Test  Проверка градиента угловой точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint4\_BoundaryPoint\_Test  Проверка градиента граничной точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint4\_IntPoint\_Test  Проверка градиента внутренней точки | True |
| MathematicialSearchPoint5  (2 ядра свертки: 3х3 вычисление градиента: максимум из градиентов) | 1 | MathematicialSearchPoint5() | MathematicialSearchPoint\_5Test  Проверка ядра свертки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint5\_CornerPoint\_Test  Проверка градиента угловой точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint5\_BoundaryPoint\_Test  Проверка градиента граничной точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint5\_IntPoint\_Test  Проверка градиента внутренней точки | True |
| MathematicialSearchPoint6  (2 ядра свертки: 5х5 вычисление градиента: по сумме квадратов) | 1 | MathematicialSearchPoint6() | MathematicialSearchPoint\_6Test  Проверка ядер свертки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint6\_CornerPoint\_Test  Проверка градиента угловой точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint6\_BoundaryPoint\_Test  Проверка градиента граничной точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint6\_IntPoint\_Test  Проверка градиента внутренней точки | True |
| MathematicialSearchPoint7 (2 ядра свертки: 7х7 вычисление градиента: по сумме квадратов) | 1 | MathematicialSearchPoint7() | MathematicialSearchPoint\_7Test  Проверка ядер свертки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint7\_CornerPoint\_Test  Проверка градиента угловой точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint7\_BoundaryPoint\_Test  Проверка градиента граничной точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint7\_IntPoint\_Test  Проверка градиента внутренней точки | True |
| MathematicialSearchPoint8  (2 ядра свертки: 5х5 вычисление градиента: по среднему арифметическому) | 1 | MathematicialSearchPoint8() | MathematicialSearchPoint\_8Test  Проверка ядра свертки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint8\_CornerPoint\_Test  Проверка градиента угловой точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint8\_BoundaryPoint\_Test  Проверка градиента граничной точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint8\_IntPoint\_Test  Проверка градиента внутренней точки | True |
| MathematicialSearchPoint9  (2 ядра свертки: 7х7 вычисление градиента: по среднему арифметическому) | 1 | MathematicialSearchPoint9() | MathematicialSearchPoint\_9Test  Проверка ядра свертки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint9\_CornerPoint\_Test  Проверка градиента угловой точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint9\_BoundaryPoint\_Test  Проверка градиента граничной точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint9\_IntPoint\_Test  Проверка градиента внутренней точки | True |
| MathematicialSearchPoint10  (2 ядра свертки: 5х5 вычисление градиента: по среднему арифметическому) | 1 | MathematicialSearchPoint10() | MathematicialSearchPoint\_10Test  Проверка ядра свертки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint10\_CornerPoint\_Test  Проверка градиента угловой точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint10\_BoundaryPoint\_Test  Проверка градиента граничной точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint10\_IntPoint\_Test  Проверка градиента внутренней точки | True |
| MathematicialSearchPoint11  (2 ядра свертки: 7х7 вычисление градиента: по среднему арифметическому) | 1 | MathematicialSearchPoint11() | MathematicialSearchPoint\_11Test  Проверка ядра свертки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint11\_CornerPoint\_Test  Проверка градиента угловой точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint11\_BoundaryPoint\_Test  Проверка градиента граничной точки | True |
| double gradientAtPoint(int x, int y) | gradientAtPoint11\_IntPoint\_Test  Проверка градиента внутренней точки | True |
| Analysis | 1 | private double getDispersion(List<double> list) | getDispersionTest Проверка вычисления дисперсии для значений, которые лежат в list вероятность каждого значения = 1/n  где n - количество элементов в list | True |
| 1 | publicList<IMathematical>getCore() | getCoreTest  Проверка возвращаемого ядра для точки, (при котором дисперсия минимальна) | True |
| 1 | public void addImageAnalysis(Image image) | addImageAnalysisTest  Проверка получения градиентов для трех разных ядер | True |
| Data | Image | 1 | Intwidth() | widthTest  Проверка получения ширины изображения в пикселях | True |
| Intheight() | heightTest()  Проверка получения высоты изображения в пикселях | True |
| Color GetPixel(int x, int y) | GetPixelTest()  Проверка получения цвета пикселя изображения | True |
| Doubletall,  Image(path) | tallTest  Проверка считывания высоты изображения при создании объекта конструктором Image(path) | True |
| Solution | 2 | void createdBeginSolution(int width, int height) | createdBeginSolution\_image10x50\_SizeBitmap10x50  Начальное решение представляет Bitmap заданного размера | True |
| void createdBeginSolution(int width, int height) | createdBeginSolution\_image10x20\_WhitePix  Начальное решение представляет Bitmap из белых пикселей | True |
| void createdBeginSolution(int width, int height) | createdBeginSolution\_image10x20\_minus1  Начальное решение представляет Bitmap с матрицей высот =-1 | True |
| ParsingInputData | Parser | 1 | Dictionary<string, double>readConfig(string path) | readConfigTest\_СorrectFile  Считывания корректного файла настроек по указанному пути | True |
| Dictionary<string, double>readConfig(string path) | readConfigTest\_InсorrectBigFile  Считывание некорректного файла настроек - количество строк больше 2-ух | True |
| Dictionary<string, double>readConfig(string path) | readConfigTest\_InсorrectSmallFile  Cчитывание некорректного файла настроек - количество строк меньше 2-ух | True |
| Dictionary<string, double>readConfig(string path) | readConfigTest\_InсorrectWayFile  Cчитывание несуществующего файла | True |
| System.Drawing.BitmapreadPNG(string path) | readPNGTest\_BlackImage Считывание изображения | True |
| Preserve | PreserveOBJ | 1 | void saveOBJ(Data.Solution solution, Data.Setting setting, string path) | savePNGTest  Проверка сохранения OBJ файла | True |
| PreservePNG | 1 | VoidsavePNGTest() | savePNGTest  Проверка сохранения изображения в png файл | True |

# Список используемых источников

1. Техническое задание на научно-исследоват
2. ельскую работу «Реконструкция 3D модели поверхности микроскопического объекта по серии изображений), Нижний Новгород, 2018.
3. Пояснительная записка № 001. Тестовый базис для тестирования ПО «Get3DModel». НИР «Get3DModel» (Тестовый базис). Н.Новгород, 2018