МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра: ИАНИ**

Магистерская программа: «Прикладная информатика»

Профиль подготовки: «Прикладная информатика в области принятия решения»

**ОТЧЕТ**

по преддипломной практике

на тему:

**«Исследование и моделирование реконструкции поверхности по серии изображений в декартовой систем координат»**

**Выполнил:** студент группы 381707-м

Губарев Сергей Юрьевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Научный руководитель:**

Чернышова Наталия Николаевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород  
2019

Оглавление

[Постановка задачи 3](#_Toc4423335)

[Алгоритм решения задачи реконструкции 3d поверхности по серии изображений 8](#_Toc4423336)

[Алгоритмы фильтрации 11](#_Toc4423337)

[Ядро свертки 17](#_Toc4423338)

[Аспекты программной реализации 22](#_Toc4423339)

[Заключение 28](#_Toc4423340)

[Список литературы 29](#_Toc4423341)

# Постановка задачи

Задачи, которые необходимо решить для достижения результатов:

1. изучить проблему предметной области
2. сформулировать задачу и определить ее место в проблемном домене
3. определить ожидаемые результаты (какой положительный эффект должен в идеале получиться).
4. формализовать задачу
5. провести анализ публичных источников, для определения уже достигнутых результатов в данной области
6. разработать и реализовать программное решение задачи
7. провести анализ полученного решения

# 

**Формализация задачи**

Суть математического решения задачи восстановления пространственного объекта по двум или более изображениям заключается в описании изменения положения камеры устройства при выполнении различных кадров. В общем случае положение определяется подъемом объектива на высоту ∆. Предполагается, что устройство камеры зафиксированные на жестком креплении, так же имеется шкала для съемки поверхности с разной высот. Расстояние от поверхности до объектива замеряется с высокой точностью. Главным считается то изображение, фото которого получено первым.

**Входные и выходные данные**

В качестве входных параметров мы имеем:

1. Параметры оптической системы, при помощи которой были получены изображения поверхности микроскопического объекта (фокусное расстояние, наблюдаемая ширина в фокусе, коэффициент для вычисления абсолютной высоты фокуса). Формат файла: файл формата .camera, имя файла – [“имя проекта”.camera], файл содержит следующие значения:

* фокусное расстояние (f=”значение параметра”)
* наблюдаемая ширина в фокусе (w=”значение параметра”)
* коэффициент для вычисления абсолютной высоты фокуса (k=”значение параметра”), по умолчанию k=1.

1. Набор изображений одинакового размера, полученный микросъемкой одного и того же объекта с разной высоты. Изображения должны содержаться в виде файлов типа PNG. Имя файлов должны быть в следующем формате: [“имя проекта”\_”относительная высота на которой было сделано изображение”.png]. Предельно допустимые размеры входных изображений от 4\*4 пикселей до 4K (4096\*3072 пикселя).

Входные данные должны содержаться в заданной системе директории при запуске.

ПО «Get3DModel» должно формировать следующие выходные данные[1]:

1. Файл формата OBJ, который будет содержать трехмерные координаты точек, записанные в стандартном формате. Имя файла: [3DModel.obj]
2. Файл формата PNG которое содержит восстановленное изображение объекта с высокой глубиной резкости. Имя файла: [sharpImage.png]

Выходные данные должны сохраняться в заданную системе директорию при запуске.

# Математическая модель

Есть серия изображений поверхности объекта с малой глубиной резкости. Изображения получены микросъемкой одного и того же объекта на разной высоте. Также имеется информация об оптической системе (фокусное расстояние, наблюдаемая ширина в фокусе, коэффициент для вычисления абсолютной высоты фокуса).

Допущение: Рассматриваются только непрозрачные объекты, а их изображения имеют одинаковый размер.

Задача: Необходимо определить координаты точек принадлежащих поверхности восстанавливаемых объектов, равномерно распределенных по исследуемой области, а также восстановить изображение объекта с высокой глубиной резкости.

Исходные параметры:

- количество изображений поверхности объекта

–размер полученных изображений: (m\*sпикселей)

–относительная высота оптической системы, на которой получено i-ое изображение,

F – фокусное расстояние

W– наблюдаемая ширина в фокусе

сoef - коэффициент для вычисления абсолютной высоты фокуса

- матрицакоординат эталонной модели.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | ... | m |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| ... |  |  |  |  |
| s |  |  |  |  |

- высота точки (z-координата) эталонной модели, соответствующая пикселям, ;

R – параметр равномерности, выраженный в процентах

Структура решения:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | ... | m |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| ... |  |  |  |  |
| s |  |  |  |  |

Решение представляет собой матрицу Z размером m\*s.Элемент матрицы:

Ограничения на решения:

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

, где

– количество уровней равномерного распределения.

Координата вектора:

,, где

–количество областейi-ого уровня, содержащих хотя бы однуточку с найденной высотой.

–количество областейi-ого уровня, на которые делим изображение.

Поэтому исходя из параметра равномерности необходимо выполнение следующего условия:

Критерий:

Рассмотрим вектора:

Пусть , тогда сточностью до обозначения:

Пусть -множество номеров компонент вектора , при которых алгоритм нашел высоты точек 3D изображения (). Оценка решения происходит на основе сравнения найденных высот 3D изображения с высотами эталонной модели.

Необходимо минимизировать модуль среднего отклонения решения от эталонной модели (при расчете не учитываются точки, чьи высоты не найдены):

# Алгоритм решения задачи реконструкции 3d поверхности по серии изображений

Алгоритм приложения «Get3DModel»

Разработанный алгоритм является итерационным, где количество итераций равно количеству входных снимков, по которым нужно построить карту глубин.

- оптимальное значение градиента для каждого пикселя на конкретной итерации;

- значение пикселя в изображении, переведенного в монохром;

- матрица высот. Изначально все элементы в ней равны -1;

**Алгоритм оценки градиента**

Шаг 1: Принимаем входные данные: снимок, сделанный на определенной высоте.

Шаг 2: Преобразуем снимок в монохромное изображение по формуле:

Шаг 3: Вычисляем «градиент» каждого пикселя, используя определенное ядро свертки G (список ядер в приложении1):

Если матрица «окружения» не может быть полностью определена для какого-либо пикселя в силу того, что это граничный пиксель изображения (пример: , тогда заполняем нулями те позиции в матрице, которые не могут быть определены.

Пример граничного пикселя и его матрицы «окружения»:

Шаг 4: Нормируем «градиент» каждого пикселя, то есть значение, полученное на шаге 3, делим на t, где ,где

G– ядро свертки размерности n.

**Алгоритм валидации точек**

В каждом ядре есть граничная ∆, 0≤ Δ ≤1 – процент порогового значения (пока равна 0), которая получена в результате работы дополнительной программы, где был найден такой процент порогового значения ∆, что сохраняя ограничение на равномерность средняя ошибка стремится к минимуму.

Порог th вычисляется так:

th = min(grad) + ( max(grad) – min(grad) )\*∆

Все градиенты, полученные алгоритмом поиска градиента на зафиксированном ядре, которые меньше полученного порога th, не рассматриваются при расчете высоты.

**Алгоритм динамического подбора ядра**

Выбор стоит между ядрами c размерностями: 3x3; 5x5; и 7x7.

Для каждого изображения в точке (x, y) вычисляются три значения градиента с помощью трех ядер разной размерности.

Обработав все изображения, для точки (x, y) мы будем иметь три списка градиентов. Каждый список соответствует одному из ядер: 3x3, 5x5, 7x7.

Чтобы определить оптимальное ядро для точки (x, y) необходимо:

Отсортировать каждый список по возрастанию.

Отбросить в каждом списке первые 2/3 значений.

Вычислить для каждого списка дисперсию.

Ядром для точки (x, y) будет являться ядро, соответствующее списку с минимальной дисперсией.

Дисперсия вычисляется по формуле:

.

**Алгоритм вычисления высоты**

Последовательно находим матрицу градиентов для каждого изображения.

Для подсчета градиента в определенном пикселе используется ядро, найденное алгоритмом динамического подбора ядра.

После расчета градиентов текущего изображения проводим операцию сравнения найденных значений матрицы градиентов с оптимальными значениями матрицы градиентов на данном этапе:

Если , то и

**Алгоритм работы программы.**

С помощью алгоритма валидации, получаем список достоверных точек.

Динамически подбираем ядра для полученных точек.

Используя найденные ядра алгоритмом динамического подбора вычисляем высоту.

**Схема оценки работы алгоритма**

Для анализа полученных выходных данных была разработана схема их оценки[2]. 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 rи 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:*

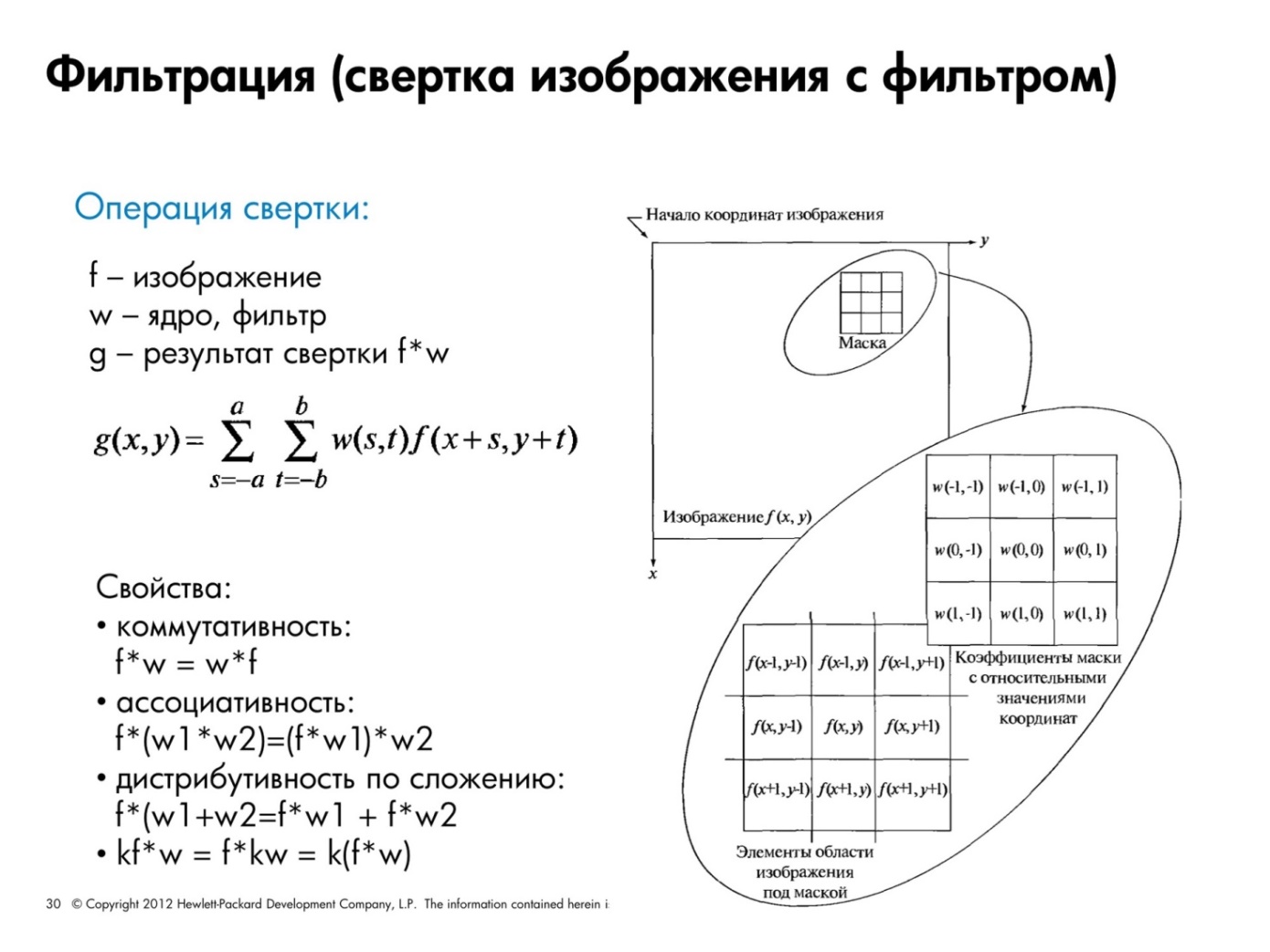
Результаты будут представлены в виде итоговой таблицы, в которой будут отражены сведения по каждому тесту отдельно, а также нормированные сведения по всем тестам.

# Алгоритмы фильтрации

Фильтрация — это применение к изображению некой функции. Операцию фильтрации называют свёрткой. Выполняется она следующим образом.

Представим, что у нас есть картинка в пространственной области и есть фильтр (он же — маска, он же — ядро) — некоторая функция. В дискретном случае это массив со значениями. Мы накладываем эту маску на кусок изображения. Тогда значение пиксела, расположенного под центральным элементом маски, вычисляется как взвешенная сумма значений пикселов, перемноженных на значения маски. То есть, накладываем маску на картинку и значение в пикселе, который под центром, вычисляется как значение пиксела, умноженное на коэффициент маски плюс значение, помноженное на коэффициент в другом месте и так далее.

Матрица свёртки – это матрица коэффициентов, которая «умножается» на значение пикселей изображения для получения требуемого результата.



Оператор Собеля

Оператор Собеля хорошо известен во всем мире и применяется для многих задач. Оператор Собеля представляет собой более неточное приближение градиента изображения, но он достаточно качественен для практического применения во многих задачах. Точнее, оператор использует значения интенсивности только в окрестности 3×3 каждого пиксела для получения приближения соответствующего градиента изображения, и использует только целочисленные значения весовых коэффициентов яркости для оценки градиент.

Оператор вычисляет градиент яркости изображения в каждой точке. Так находится направление наибольшего увеличения яркости и величина её изменения в этом направлении. Результат показывает, насколько «резко» или «плавно» меняется яркость изображения в каждой точке, а значит, вероятность нахождения точки на грани, а также ориентацию границы.

Т.о. результатом работы оператора Собеля в точке области постоянной яркости будет нулевой вектор, а в точке, лежащей на границе областей различной яркости — вектор, пересекающий границу в направлении увеличения яркости.

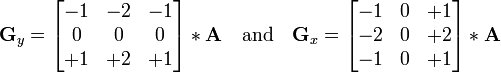
Формула оператора Собеля:

и — две матрицы, где каждая точка содержит приближенные производные по x и по y. Они вычисляются следующим образом путем умножения матрицы и и суммированием обоих матриц, в результате полученный результат записывается в текущие координаты x и y в новое изображение:

Наиболее часто оператор Собеля применяется в алгоритмах выделения границ.

image

Матрицы и :



где \* обозначает двумерную операцию свертки.

Оператор Прюитта

Алгоритм оператора Прюитта подобен алгоритму оператору Собеля, за исключением использования другой матрицы:

Для этой операции используются различные ядра. Из одного ядра можно получить восемь, переставляя вращательно коэффициенты. Каждый результат будет чувствителен к направлению границы от 0° до 315° с шагом в 45°, где 0° соответствует вертикальной границе.

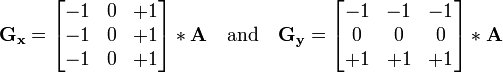
Максимальный ответ каждого пикселя есть значение соответствующего пикселя в выходном изображении. Значения его лежат между 1 и 8, в зависимости от номера ядра, давшего наибольший результат.

Этот метод выделения границ также называется подстановкой шаблонов границ, поскольку изображению сопоставляется набору шаблонов, и каждый представляет некоторую ориентацию границы. Величина и ориентация границы в пикселе тогда определяется шаблоном, который лучше всех соответствует локальной окрестности пикселя.

Детектор границ Прюитт является подходящим способом для оценки величины и ориентации границы. В то время как детектор с дифференциальным [градиентом](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%82) нуждается в трудоёмком вычислении оценки ориентации по величинам в вертикальном и горизонтальном направлениях, детектор границ Прюитт даёт направление прямо из ядра с максимальным результатом. Набор ядер ограничен 8 возможными направлениями, однако опыт показывает, что большинство прямых оценок ориентации тоже не очень точны.

С другой стороны, набор ядер нуждается в 8 свёртках для каждого пикселя, тогда как набор ядер градиентного метода требует только 2: чувствительных по вертикали и по горизонтали. Результат для изображения мощности границ очень похож у обоих методов, если в них используются те же ядра свёртки.

Оператор использует два ядра 3×3, свёртывая исходное изображение для вычисления приближённых значений производных: одно по горизонтали и одно по вертикали. Положим A исходным изображением, и G{x},G{y}— двумя изображениями, в которых каждая точка содержит горизонтальное и вертикальное приближение производной, которая рассчитывается как:



где \* обозначает двумерную операцию свертки.

Перекрёстный оператор Робертса

Перекрёстный оператор Робертса — один из ранних алгоритмов выделения границ, который вычисляет сумму квадратов разниц между диагонально смежными пикселами. Это может быть выполнено сверткой изображения с двумя ядрами:

image

Иными словами, величина перепада G получаемого изображения вычисляется из исходных значений параметра Y в дискретных точках растра с координатами (x,y) по правилу:

(для Евклидовой метрики, но иногда в прикладных случаях модуль вектора перепада в методе Робертса может ускоренно вычисляться в Метрике городских кварталов ). То есть в методе Робертса используется суммарный вектор из двух диагональных векторов перепада. И в операторе Робертса используется модуль этого суммарного вектора, который показывает наибольшую величину перепада между четырьмя охваченными точками. А направление этого вектора соответствует направлению наибольшего перепада между точками (в статье он не описан, но тоже находит применение в анализе картины двумерного распределения параметра Y).

Преобразование каждого пикселя перекрёстным оператором Робертса может показать производную изображения вдоль ненулевой диагонали, и комбинация этих преобразованных изображений может также рассматриваться как градиент от двух верхних пикселов к двум нижним. Оператор Робертса всё ещё используется ради быстроты вычислений, но он проигрывает в сравнении с альтернативами из-за значительной чувствительности к шуму, что часто неприемлемо. Он даёт линии тоньше, чем другие методы выделения границ, что почти равносильно вычислению конечных разностей вдоль координат X и Y. Иногда его называют «фильтром Робертса».

На картине двумерного распределения в качестве параметра Y обычно выступают значения любых полей, например, яркость цветового канала, интенсивность излучения, температура или т.п.

В результате использования операции дискретного двумерного дифференцирования получается новое значение, которое записывается в новое изображение.

Оператор Лапласа

Дискретный оператор Лапласа часто используется в обработке изображений, например в задаче выделения границ или в приложениях оценки движения. Дискретный лапласиан определяется как сумма вторых производных и вычисляется как сумма перепадов на соседях центрального пикселя.

Можно заметить, что вторая производная , в точке максимума первой, равна нулю. Таким образом, мы можем также использовать этот критерий, чтобы попытаться обнаружить контуры в изображении. Тем не менее, значение производной может обращаться в нуль и в других точках. Эта проблема может быть решена путем предварительной фильтрации изображения. Из приведенного выше объяснения, получаем, что вторая производная может быть использована для обнаружения контура на изображении. Когда мы говорим про изображение, мы рассматриваем "2D" пространство. Метод усиления края по Лапласу рассматривает целый ряд различных ядер свертки. Приведем некоторые их них:

|  |  |  |  |
| --- | --- | --- | --- |
| https://studfiles.net/html/1144/349/html_htR7gvBXGZ.G72d/img-do6AP5.png | https://studfiles.net/html/1144/349/html_htR7gvBXGZ.G72d/img-luuloG.png | https://studfiles.net/html/1144/349/html_htR7gvBXGZ.G72d/img-hktoi9.png | (1.22) |

Как видно, сумма элементов матриц равна нулю, поэтому отклик фильтра может быть отрицательным. В этом случае значение отклика берется по модулю. В результате обработки области с постоянной или линейно возрастающей интенсивностью становятся черными, а области быстро изменяющихся значений интенсивности ярко высвечиваются.

Ниже приведем некоторые пространственные процессы, которые не подпадают под категорию свертки и могут применяться для устранения различного вида шума.



# Ядро свертки

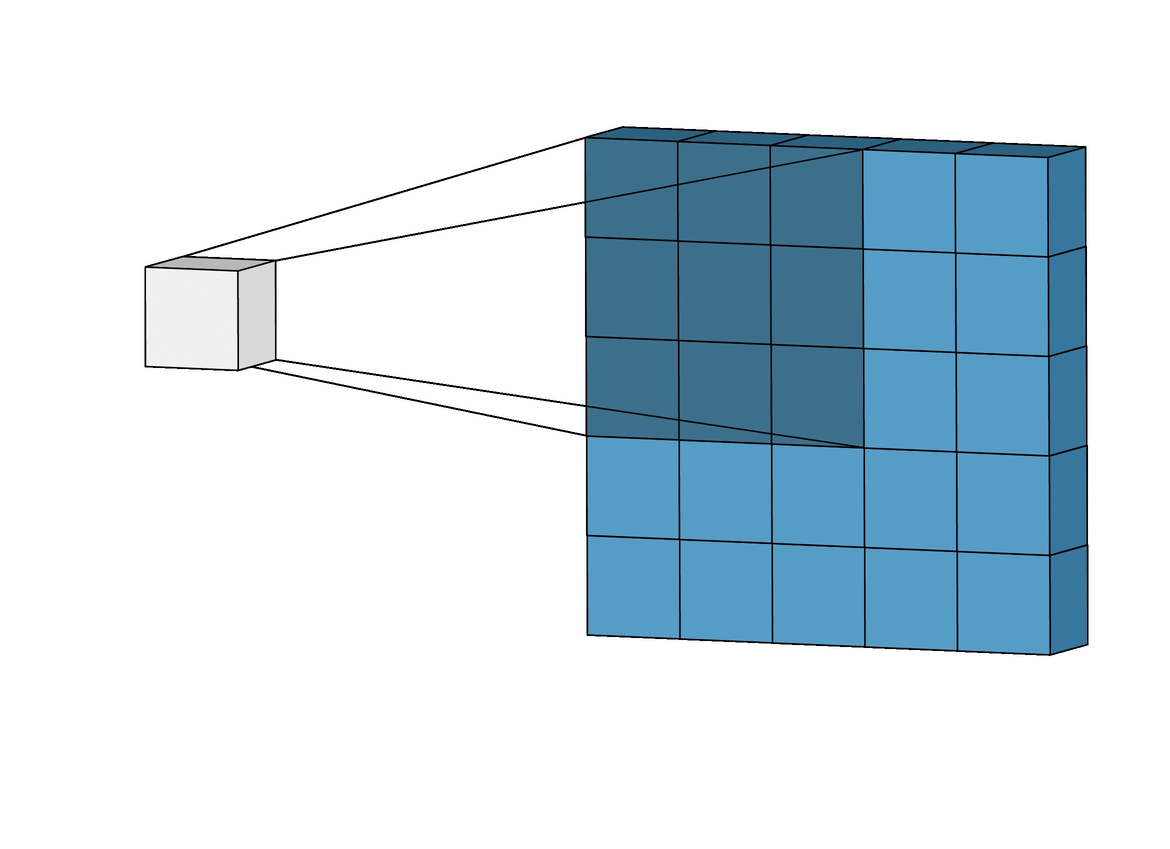
Свертка (англ. convolution) — это операция, показывающая «схожесть» одной функции с отражённой и сдвинутой копией другой. Cвертка – это операция вычисления нового значения выбранного пикселя, учитывающая значения окружающих его пикселей. Для вычисления значения используется матрица, называемая *ядром свертки*. Обычно ядро свертки является квадратной матрицей n\*n, где n — нечетное, однако ничто не мешает сделать матрицу прямоугольной. Во время вычисления нового значения выбранного пикселя ядро свертки как бы «прикладывается» своим центром (именно тут важна нечетность размера матрицы) к данному пикселю. Окружающие пиксели так же накрываются ядром. Далее высчитывается сумма, где слагаемыми являются произведения значений пикселей на значения ячейки ядра, накрывшей данный пиксель. Сумма делится на сумму всех элементов ядра свертки. Полученное значение как раз и является новым значением выбранного пикселя. Если применить свертку к каждому пикселю изображения, то в результате получится некий эффект, зависящий от выбранного ядра свертки.

**Свертка** (англ. *convolution*) — операция над парой матриц AA (размера nx×nynx×ny) и BB (размера mx×mymx×my), результатом которой является матрица C=A∗BC=A∗B размера (nx−mx+1)×(ny−my+1)(nx−mx+1)×(ny−my+1). Каждый элемент результата вычисляется как скалярное произведение матрицы BB и некоторой подматрицы AA такого же размера (подматрица определяется положением элемента в результате). То есть, Ci,j=∑mx−1u=0∑my−1v=0Ai+u,j+vBu,vCi,j=∑u=0mx−1∑v=0my−1Ai+u,j+vBu,v. На изображении справа можно видеть, как матрица BB «двигается» по матрице AA, и в каждом положении считается скалярное произведение матрицы BB и той части матрицы AA, на которую она сейчас наложена. Получившееся число записывается в соответствующий элемент результата.

Логический смысл свертки такой — чем больше величина элемента свертки, тем больше эта часть матрицы AA была похожа на матрицу BB (похожа в смысле скалярного произведения). Поэтому матрицу AAназывают *изображением*, а матрицу BB — *фильтром* или *образцом*.

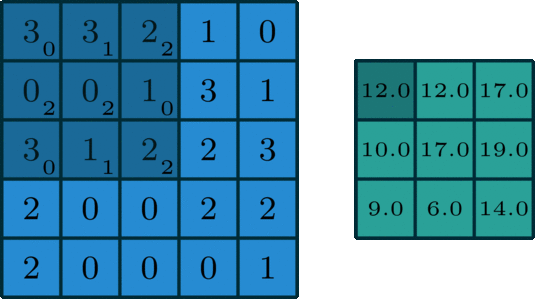
Что в свою очередь позволяет работать с более малой матрицей но столь же информативной (с точки зрения исходной матрицы). Это значительно отражается на времени обработки данного изображения.

**Двумерная сверточная нейронная сеть**



**Двумерная свертка (2D convolution)** — это довольно простая операция: начинаем с ядра, представляющего из себя матрицу весов (weight matrix). Ядро “скользит” над двумерным изображением, поэлементно выполняя операцию умножения с той частью входных данных, над которой оно сейчас находится, и затем суммирует все полученные значения в один выходной пиксель.

Ядро повторяет эту процедуру с каждой локацией, над которой оно “скользит”, преобразуя двумерную матрицу в другую все еще двумерную матрицу признаков. Признаки на выходе являются взвешенными суммами (где веса являются значениями самого ядра) признаков на входе, расположенных примерно в том же месте, что и выходной пиксель на входном слое.



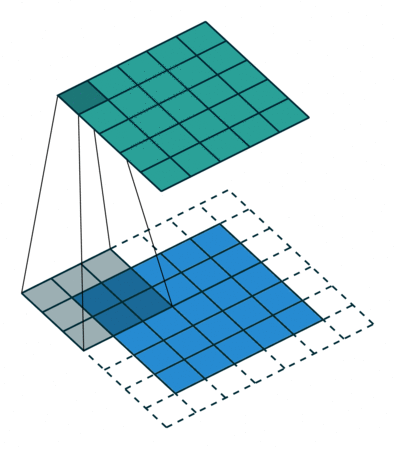
Операция свертки

Независимо от того, попадает ли входной признак в “примерно то же место”, он определяется в зависимости от того, находится он в зоне ядра, создающего выходные данные, или нет. Это значит, что **размер ядра свертки определяет количество признаков, которые будут объединены для получения нового признака на выходе.**

В примере, приведенном выше, мы имеем 5\*5=25 признаков на входе и 3\*3=9 признаков на выходе. Для стандартного слоя мы бы имели весовую матрицу 25\*9 = 225 параметров, а каждый выходной признак являлся бы взвешенной суммой всех признаков на входе. Свертка позволяет произвести такую операцию с всего 9-ю параметрами, ведь каждый признак на выходе получается анализом не каждого признака на входе, а только одного входного, находящегося в “примерно том же месте”. Обратите на это внимание, так как это будет иметь важное значение для дальнейшего обсуждения.

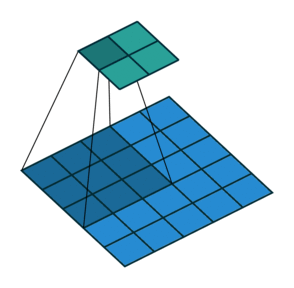
**Часто используемые техники**

Перед тем как мы двинемся дальше, безусловно стоит взглянуть на две техники, которые часто применяются в сверточных нейронных сетях: Padding и Striding.



**Padding.** Если вы наблюдаете анимацию, обратите внимание на то, что в процессе скольжения края по существу обрезаются, преобразуя матрицу признаков размером 5\*5 в матрицу 3\*3. Крайние пиксели никогда не оказываются в центре ядра, потому что тогда ядру не над чем будет скользить за краем. Это совсем не идеальный вариант, так как мы хотим, чтобы размер на выходе равнялся входному.

[Padding](https://arxiv.org/abs/1603.07285) добавляет к краям поддельные пиксели (обычно нулевого значения, вследствие этого к ним применяется термин “нулевое дополнение” — “zero padding”). Таким образом, ядро при проскальзывании позволяет неподдельным пикселям оказываться в своем центре, а затем распространяется на поддельные пиксели за пределами края, создавая выходную матрицу того же размера, что и входная.



Свертка с шагом 2

**Striding.** Часто бывает, что при работе со сверточным слоем, нужно получить выходные данные меньшего размера, чем входные. Это обычно необходимо в сверточных нейронных сетях, где размер пространственных размеров уменьшается при увеличении количества каналов. Один из способов достижения этого — использование субдискритизирующих слоев (pooling layer), например, принимать среднее/максимальное значение каждой ветки размером 2\*2, чтобы уменьшить все пространственные размеры в два раза. Еще один способ добиться этого — использовать stride (шаг).

Идея stride заключается в том, чтобы пропустить некоторые области, над которыми скользит ядро. Шаг 1 означает, что берутся пролеты через пиксель, то есть по факту каждый пролет является стандартной сверткой. Шаг 2 означает, что пролеты совершаются через каждые два пикселя, пропуская все другие пролеты в процессе и уменьшая их количество примерно в 2 раза, шаг 3 означает пропуск 3-х пикселей, сокращая количество в 3 раза и т.д.

# Аспекты архитектуры информационной системы

Под решение данной задачи, очевидна необходимость разработки программного обеспечения: «Get3DModel» в рамках текущей работы предназначенное для получения 3D модели по серии изображений поверхности объекта, полученных оптической системой с малой глубиной резкости на разной высоте.

**Высокоуровневый дизайн программы**

Дизайн высокого уровня - понятие, используемое для характеристики архитектуры, которая будет использоваться для создания программного продукта. Архитектурная схема представляет собой обзор всей системы, выявление основных компонентов, которые будут разработаны для продуктов и их интерфейсов. ДВР использует от возможных нетехнических к технических терминов, которые должны быть понятными для администраторов системы. В отличие от [низкого уровня дизайна](https://uk.wikipedia.org/wiki/%D0%9D%D0%B8%D0%B7%D1%8C%D0%BA%D0%BE%D1%80%D1%96%D0%B2%D0%BD%D0%B5%D0%B2%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) , который дополнительно раскрывает логическую подробную конструкцию каждого из этих элементов для программистов. Дизайн высокого уровня должен кратко описать все платформы, системы, продукты, услуги и процессы, от которых зависит, и включать любые важные изменения, которые нужно внести. Так же должен иметь полный пересмотр всей системы, разбивая ее на более мелкие детали, которые легче понять. Чтобы свести к минимуму накладные расходы на техническое обслуживание по мере продолжения строительства и проектирования нижнего уровня, лучше всего, чтобы проект верхнего уровня разрабатывался только в той степени, которая необходима для удовлетворения этих потребностей.

При построении 3D модели поверхности, непрозрачного объекта из предоставленной серии изображений и параметров оптической системы, результатами которого будут координаты точек, записанных в файле, форма OBJ, а также восстановленное изображение объекта с высокой глубиной резкости – файл формата PNG.

Для реализации информационной системы было выделено 4 основных компонента:

Parser – Блок отвечающий за чтение и преобразование изображения, а так же чтение файла настроек.  
Функционал:

|  |  |  |
| --- | --- | --- |
|  | Вход | Выход |
| Прочитать файл настроек | Путь к файлу настроек | Массив значений  [Имя настройки, значение] |
| Прочитать и преобразовать изображение | Путь к изображению | Преобразованное изображение - матрица цветов |

Calculatedblock – Расчетный блок. Отвечает за построение по преобразованным изображениям матрицу высот, где [x,y] – координаты пикселя, а значение – относительная высота на которой сделано изображение. Так же расчетный блок строит изображение высокой резкости.

Функционал:

|  |  |  |
| --- | --- | --- |
|  | Вход | Выход |
| Создать начальное решение |  | Созданное решение |
| Уточнить решение | Преобразованное изображение, относительная высота на которой сделано изображение | Уточнённое решение |
| Забрать решение |  | Полученное решение расчетного блока |

Под «решением» будем понимать матрицу высот и преобразованное изображение высокой резкости.

PreservePNG – Блок отвечающий за преобразование изображения высокой резкости и сохранения его в png файл.  
Функционал:

|  |  |  |
| --- | --- | --- |
|  | Вход | Выход |
| Преобразовать и сохранить изображение | Преобразованное изображение для последующего сохранения | PNG файл |

PreserveOBJ – блок отвечающий за расчет матрицы высот и ее преобразование согласно исходным настройкам оптической системы, полученной ранее, с последующие сохранением результатов в файл формата obj.

Функционал:

|  |  |  |
| --- | --- | --- |
|  | Вход | Выход |
| Рассчитать и сохранить obj файл | Полученное решение расчетного блока и настройки оптической системы | OBJфайл |

Связь компонентов отображена на рис.1

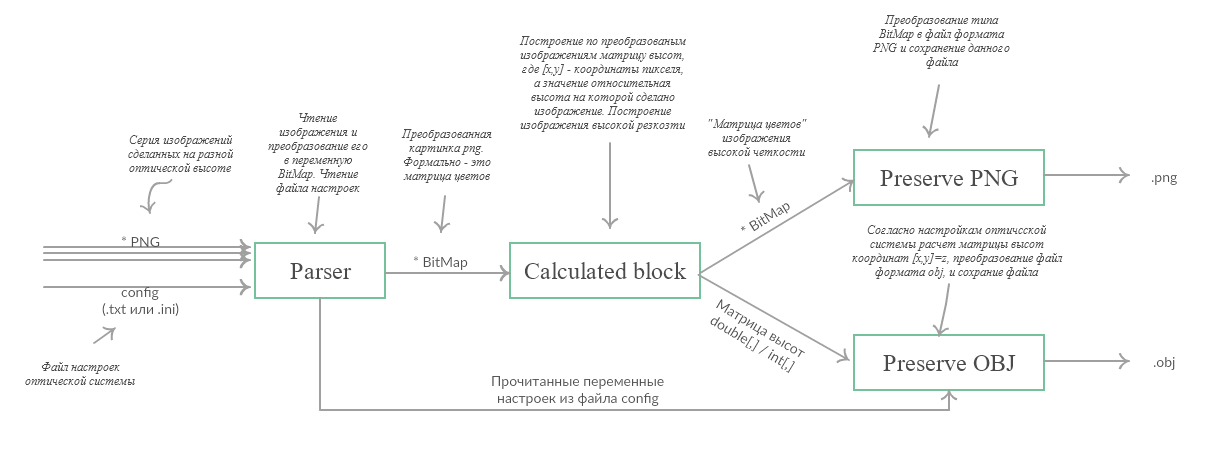


рис. 1. Высокоуровненый дизайн системы.

Серия изображений, сделанных на разной высоте и файл с параметрами оптической системы поступают на вход блоку Parser, который читает данные и преобразует данные в матрицу цветов. Эта матрица и относительная высота, на которой сделано изображение являются входными данными для следующего расчетного блока Calculatedblock, который создает начальное решение - матрица высот и преобразованное изображение высокой резкости, а в дальнейшем уточняет его. Далее PreservePNG преобразованное изображение высокой резкости сохраняет изображение в формате .png, а PreserveOBJ полученное решение расчетного блока и настройки оптической системы преобразует и сохраняет в файл формата .obj.

Ход выполнения программы изображен на рис.2

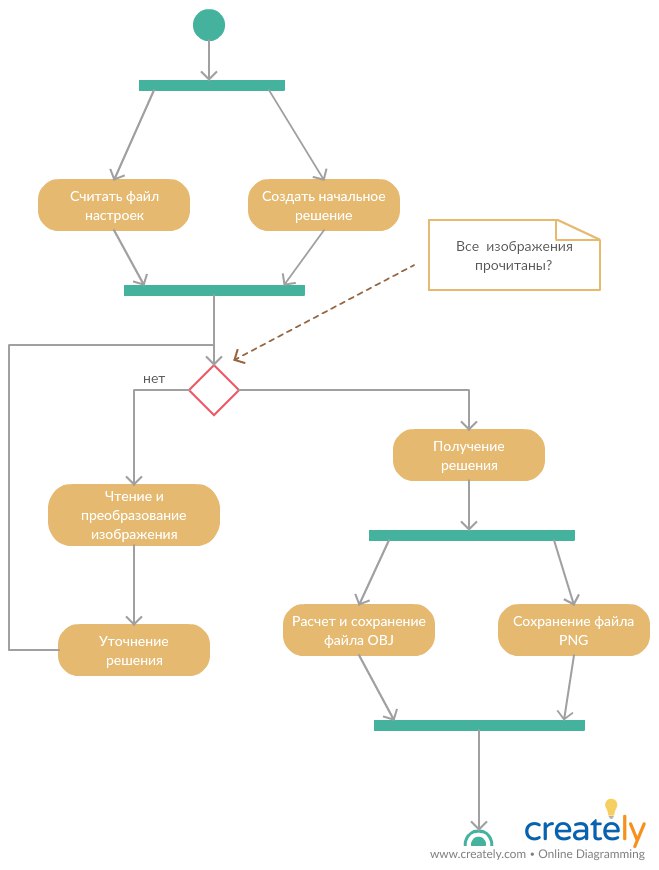


рис. 2. Модель взаимодействия с программной системой.

**Аспекты реализации расчетного блока Calculateblock**

Рассматриваемый блок состоит из 5 логических блоков (UML-диаграмма представлена в Приложении 2):

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 классовнаследуемых 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. Получить решение  
      Solution getSolution();

# Заключение

В результате исследования в данной области «Реконструкция 3D модели поверхности микроскопического объекта по серии изображений», был достигнуты следующие результаты:

1. Исследована предметная область
2. Сформулирована задача
3. Построена математическая модель
4. Исследован и разработан алгоритм решения задачи
5. Разработана архитектура приложения
6. Подготовлен тестовый базис
7. Разработана тестовая инфраструктура
8. Проведен анализ полученных результатов: найдена оптимальная комбинация параметров

В результате проделанной работы было разработано ПО, с помощью которого могут быть получены координаты точек в трехмерном пространстве, записанных в файле форма OBJ и восстановленное изображение объекта с высокой глубиной резкости – файл формата PNG.

# Список литературы

1. Агапов И.А., Кашкин В.Б. Обработка изображений: метод. указания, ч. 1,2//Красноярский гос.ун-т, Красноярск, 1994.
2. Гонсалес Р., Вудс Р. Цифровая обработка изображений. — М.: Техносфера, 2006. — 1072 с.
3. Яне Б. Цифровая обработка изображений. — М.: Техносфера, 2007. — 584 с