МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

"Ярославский государственный университет им. П.Г.Демидова".

Кафедра компьютерных сетей

   «Допустить к защите»

Профессор,

          д-р. ф-м. наук,

              \_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.Д. Глызин,

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

**Курсовая работа**

по специальности 01.03.02 Прикладная математика и информатика

**Улучшение качества растровых изображений с помощью генетического алгоритма.**

  Научный руководитель

Доцент,

         канд. ф-м. наук,

             \_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.В. Краснов

        Студент группы ИВТ-32БО

      \_\_\_\_\_\_\_\_\_\_\_\_ Ющенко А. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Ярославль 2019 г.

**Содержание**

План:

1 Анализ сущ реш

2 Растравое изображение

3 Методы (операторы) улучшения изобр

4 Ген алгоритм (много подпунктов)

5 Оптимизация

Введение

Создание снимков высокого качества является необходимым во многих производственных и научных областях. Очень часто встречаются такие ситуации, когда условия съемки не позволяют добиться приемлемого качества изображения или получение высококачественного изображение достаточно дорого, например, фотографии с марсохода, или же подводного батискафа. Так же эта задача часто встречается в качестве промежуточного звена в других разделах науки. Одним из ярчайших примеров является компьютерное зрение. Поэтому, на сегодняшний день, улучшение существующих изображений является более чем актуальной задачей.

На данный момент существует огромное количество методов улучшение изображений, однако большинство из них являются эвристическими, то есть решают какую-то специфическую задачу. Так же почти все эти методы требуют человеческого взаимодействия, для настройки некоторых параметров. Целью моей работы являлось создания автоматического, объективного метода улучшения изображений в основе которого лежит генетический алгоритм.

**Глава 1. Анализ существующих решений и постановка задачи**

**1.1 Анализ существующих работ**

# Я начал свое ознакомление с работы Махно Таисия Александровны, которая посвящена обработке медицинских изображений с помощью эволюционных алгоритмов. Из этой работы подчерпнул достаточно много полезной информации касающейся генетического алгоритма, однако более полезным оказалась для меня работа А.А. Белоусова, В.Г. Спицына, Д.В. Сидорова, которая посвящена какой же проблеме как у меня. К сожалению, мне не удалось найти реализацию их алгоритма, так же большое число технических деталей было опущено, но теоритическая часть, касающаяся оценки изображений оказалась крайне полезной.

# 1.2 Постановка задачи

Как я уже говорил выше цель моей работы разработать метод для улучшение растровых изображений, то есть на вход поступает изображение, после чего оно должно быть оценено и на основе этой оценки генетический алгоритм должен подобрать веса для фильтров и на выходе получить оптимальное по резкости и контрастности изображение.

Были поставленный следующие задачи:

1. Найти и запрограммировать подходящие фильтры и операторы, с помощью которых будет осуществляться улучшение изображений.

2. Построить модель генетического алгоритма.

3. Найти и запрограммировать алгоритмы без эталонной оценки изображений.

4. Ускорить и оптимизировать решение.

**Глава 2. Растровые изображения**

Для того что бы работать с растровыми изображениями на отчетливо понимать, что они из себя представляют и какие операции над ними можно производить.

Растровые изображения это один из наиболее популярных способов хранения изображения на компьютере. Он представляет из себя массив в котором каждые 3 или 4 ячейки кодируют цвет одного пикселя. Каждое значение обозначает яркость цветового канала, то есть если цвет кодируется тремя ячейками, то первая отвечает за красный канал, вторая за зеленый и последняя за синий. Некоторые форматы содержат еще и четвертую компоненту, которая отвечает за альфа канал и нужна для смешивания при наложении двух изображений друг на друга, в данной работе она не используется. Выше было описано цветовое пространство RGB, однако существуют и другие, о них речь пойдет позже. Ну и последним понятием является глубина цветового канала, или то сколько значений имеет цветовой канал. Самым популярным является кодирование цветового канала одним байтом, то есть каждый канал принимает значения в диапазоне от 0 до 255, где 255 означает что влияние цветового канала максимально. То есть последовательность (255, 0, 0) кодирует красный цвет. Еще одним из популярных кодирований является 4 байта на канал. Легко заметить, что изображения в этом случаи будут занимать в четыре раза больше памяти, однако появляется возможность хранить отрицательные значения что является необходимым для некоторых цветовых пространств.

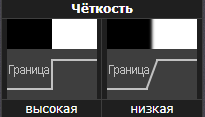
Еще несколько полезных определений:

Яркость пикселя - это среднее арифметическое всех каналов, кроме альфа канала.

Контрастность изображения – это разница в яркости некоторых участков фотографии. Если говорить очень просто, то самые темные участки изображения должны быть черными, а самые светлые – белыми, иначе изображение выглядит бледно.

Вот пример для наглядности:



Резкость изображения – это характеристика изображения которая описывает различимость деталей на фотографии. На воспринимаемую резкость изображения влияют два фундаментальных фактора: разрешение и чёткость.

В готовом изображении я могу оперировать только четкостью, подробнее про это будет в следующей главе.

Пример:

**Глава 3. Алгоритмы улучшения изображения**

Для того что бы разобраться с основной задачей необходимо понять, как мы вообще можем улучшить изображение. Я разберу алгоритмы которые я использовал в конечной программе.

**Гистограмма**

Для начала введу такое важное понятие как гистограмма.

Гистограмма — это график [статистического](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0) распределения элементов цифрового изображения с различной [яркостью](https://ru.wikipedia.org/wiki/%D0%AF%D1%80%D0%BA%D0%BE%D1%81%D1%82%D1%8C), в котором по горизонтальной оси представлена яркость, а по вертикали — относительное число [пикселей](https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D0%BA%D1%81%D0%B5%D0%BB%D1%8C) с конкретным значением яркости.

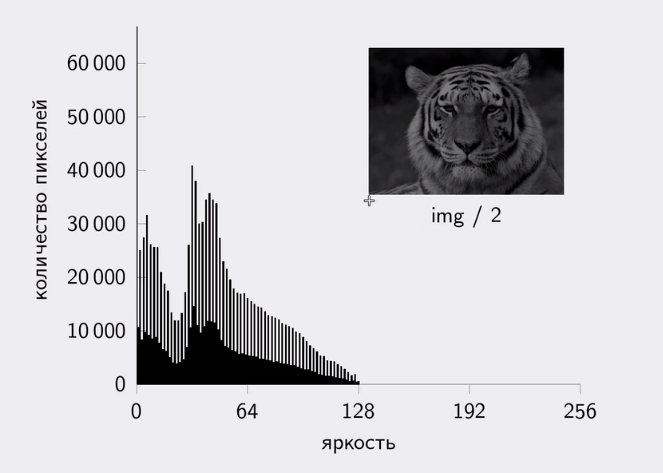
Изучив гистограмму, можно получить общее представление о правильности [экспозиции](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D1%81%D0%BF%D0%BE%D0%B7%D0%B8%D1%86%D0%B8%D1%8F_(%D1%84%D0%BE%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)), [контрасте](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%B0%D1%81%D1%82) и цветовом насыщении снимка, оценить требуемую коррекцию как при съёмке (изменение экспозиции, цветового баланса, освещения либо композиции снимка), так и при последующей обработке. Гистограмма может быть, как и для общей яркости, так и для яркости каждого канало по отдельности. Забегая вперед, скажу, что все преобразования будут касаться только яркости.

Для получения гистограммы достаточно просто пройтись по всем пикселям и посчитать их сумму для каждого значения яркости пикселя и яркости каждого канала.

Значения полученной таблицы будет использоваться в алгоритмах далее.

Гистограмма изображения выглядит обычно так:

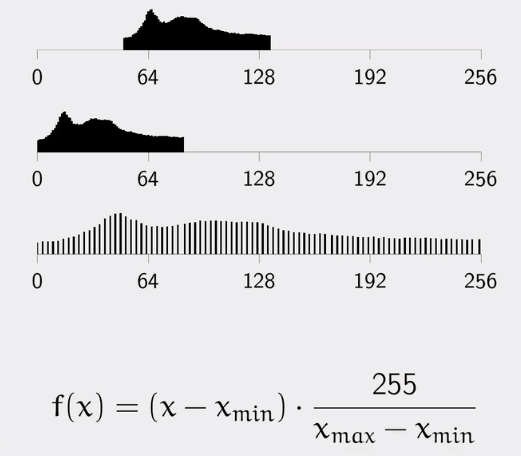
**Линейное растяжение гистограммы**

Итак, первое преобразование которое мы рассмотрим называется линейным растяжением гистограммы. Так как у слабо контрастного изображения яркость всех пиксели сосредоточены в одной области, то достаточно очевидным решением является изменить яркость всех пикселей так чтобы они заполняли весь диапазон яркости.

Для этого нам нужно сместить все пиксели влево так чтобы был хотя бы один черный и умножить на коэффициент. Коэффициент выводиться очевидным образом. Нам нужно чтобы достигалась максимальная яркость или x\_max = 255. И так же минимальная x\_min = 0.

(x\_max – x\_min) \* 255 / (x\_max – x\_min) = 255.

(x\_min – x\_min) \* 255 / (x\_max – x\_min) = 0.

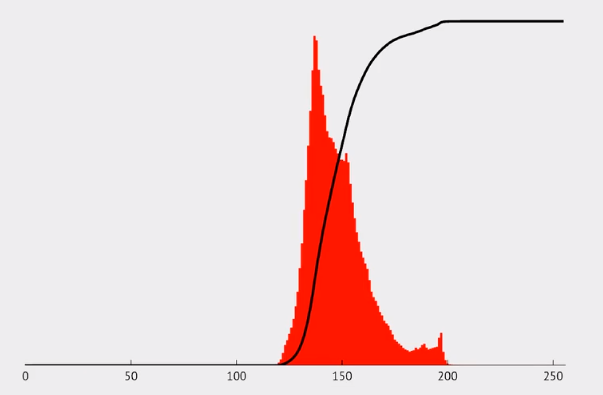
Таким образом получаем такую формулу.

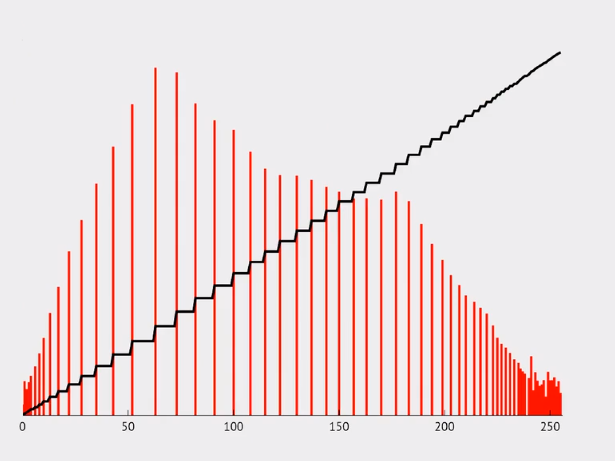
Эта формула достаточна эффективна, однако, с изображениями такого вида она не справиться.

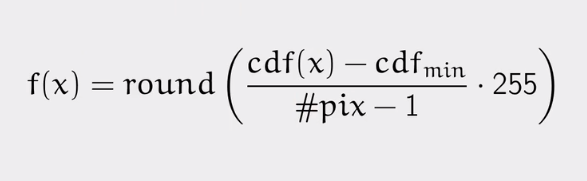


Поэму введем дополнительный параметр который будет отвечать за количество пикселей на краях, которые мы не будем учитывать. Данный метод является уже вполне рабочим, и я его использовал в своей работе.

**Выравнивание гистограммы**

 Как понятно из названия данный метод, как и прошлый, работает с гистограммой. Для начала рассмотрим гистограмму слабо контрастного изображения.

График изображенный поверх гистограммы является функцией распределения. Суть данного метода заключается в том, чтобы функция распределения гистограммы итогового изображения была похожа на линейную. Результат должен выглядеть примерно так.

Итак, в этот раз формула будет выглядеть сложнее, но суть остается та же, заполнить все яркостное пространство изображения.

Где #pix – количество пикселей, а cdf(x) значение функции распределения в точке x. Функция распределения выглядит так.

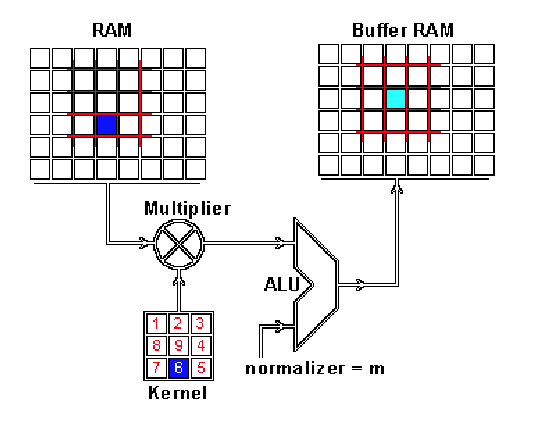
Где h(x) – количество пикселей с яркостью x.

Сравнение двух алгоритмов:

Сверху изначальное изображение, слева Выравнивание гистограмм, справа Авто контраст. В данном случаи Выравнивание гистограмм заметно выигрывает, однако это происходит далеко не всегда, к тому же у Авто контраста есть параметр (кол-во пропускаемых самых светлых и самых темных пикселей) который дает кое какую гибкость.

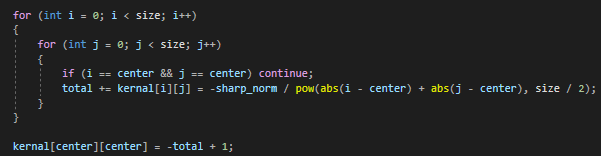
**Свертки**

Теперь поговорим об операциях свертки – еще одном эффективном преобразовании изображений.

Операция свертки – это применение ядра свертки для каждого пикселя изображения. Ядро представляет собой матрицу размера (2N+1)x(2N+1) с различными весами. Мы как бы накладываем нашу матрицу поверх изображения и считаем сумму пикселей, умноженных на соответствующие веса в матрице, и результат записываем в тот пиксель где был цент матрицы.

Так же для того чтобы яркость изображения оставалась неизменной, сумма всех весов в ядре должна равняться единице. Для пикселей которые выходят за края изображения берутся значения, отзеркаленные от границы.

**Фильтр резкости**

Оператор свертки у ядра которого все значения отрицательные кроме центрального, «выделяет» пиксель на фоне соседних, тем самым повышает резкость изображения. У ядра можно изменять как размер, так и веса. Увеличение ядра, ведет за собой увеличение времени применение, мне достаточно ядра с фиксированным размером я выбрал 3х3 и параметром sharp, создание ядра я реализовал таким образом.

После применяя свертки с таким ядром, резкость изображения увеличится в зависимости от входного параметра sharp, это как раз то что мне и нужно.

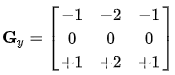
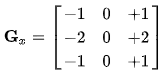
**Box фильтр**

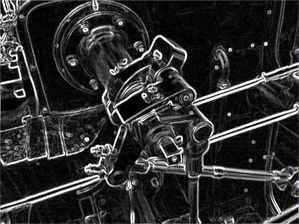
В противовес фильтру резкости я добавил фильтр размытия. Усредняющий фильтр или Box фильтр размера 3x3 имеет одинаковые веса равные 1/9. Иногда бывали ситуации что после последовательности, фильтра размытия и фильтра резкости качество улучшалось.

**Оценка изображения**

Для того что бы использовать генетический алгоритм нам нужно каким-то образом оценивать изображения. Общего алгоритма оценки изображений не существует так как разные категории фотографий имеют различные критерии и меры точной оценки, поэтому в случаи, когда нет эталона приходиться руководствоваться разными эвристиками и общими понятиями присущими всем изображениям. Например, подойдут контрастность и резкость.

**Оценка контрастности изображения**

Контрастность — это разница между глубиной темных областей и светлых. Таким образом контрастность можно измерить через количество и интенсивность краевых пикселей. Улучшенное изображение имеет большее число краевых пикселей с более высокими значениями интенсивностей по сравнению с исходным изображением. Для нахождения краевых пикселей я воспользуюсь оператором Собеля, который представляет собой ядро свертки 3х3, я использовал среднее арифметическое между двумя полученными градиентами изображения по x и по y. Ядра выглядят так.

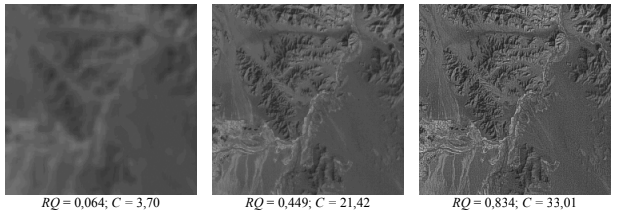
После применения данных сверток мы получим примерно такое изображение.

Будет достаточно просто пройтись по всем пикселям и посчитать суммарную яркость, это и будет оценкой контрастности.

**Оценка резкости изображения**

Для оценки я воспользовался результатами другой рабаты и просто взял формулу, которая выглядит так.

Где f (i, j) – значение яркости пикселя с координатами (i, j), K – половина максимального значения яркости, M, N – ширина и высота изображения. На этой фотографии видны значения оценки RQ для изображений с разной резкостью.



**Глава 4. Генетический алгоритм**

Генетические алгоритмы предназначены для решения задач оптимизации. Примером подобной задачи может служить обучение нейросети, то есть подбора таких значений весов, при которых достигается минимальная ошибка. При этом в основе генетического алгоритма лежит метод случайного поиска. Основным недостатком случайного поиска является то, что нам неизвестно сколько понадобится времени для решения задачи. Для того, чтобы избежать таких расходов времени при решении задачи, применяются методы, проявившиеся в биологии. При этом используются методы открытые при изучении эволюции и происхождения видов. Как известно, в процессе эволюции выживают наиболее приспособленные особи. Это приводит к тому, что приспособленность популяции возрастает, позволяя ей лучше выживать в изменяющихся условиях.

**Представление объектов**

Из биологии мы знаем, что любой организм может быть представлен своим фенотипом, который фактически определяет, чем является объект в реальном мире, и генотипом, который содержит всю информацию об объекте на уровне хромосомного набора. При этом каждый ген, то есть элемент информации генотипа, имеет свое отражение в фенотипе. Таким образом, для решения задач нам необходимо представить каждый признак объекта в форме, подходящей для использования в генетическом алгоритме. Все дальнейшее функционирование механизмов генетического алгоритма производится на уровне генотипа, позволяя обойтись без информации о внутренней структуре объекта, что и обуславливает его широкое применение в самых разных задачах.

В наиболее часто встречающейся разновидности генетического алгоритма для представления генотипа объекта применяются битовые строки. При этом каждому атрибуту объекта в фенотипе соответствует один ген в генотипе объекта. Ген представляет собой битовую строку, чаще всего фиксированной длины, которая представляет собой значение этого признака.

Моя особь состоит из 2 генотипов фиксированной длины равной 3. Первый отвечает тип, второй за значение аргумента. Первое место отведено под операторы растяжение гистограмм, второй и третий под фильтры. Такое количество и порядок обосновывается моим наблюдением, и оптимизацией. Изначально цепочки были не ограничены, а любые операторы могли стоять в любом месте и повторяться. Такой подход был не оптимален со всех точек зрения, скорости и эффективности. Однако я заметил некоторую закономерность для всех изображений, собственно хороший результат был только у цепочек с генотипом который я описал выше. Его оказалось более чем достаточно для улучшения изображений, так же генетический алгоритм стал на 2 порядка быстрее, так что простота в этом случаи принесла только выигрыш.

Итак, значение первого гена отвечает за выравнивание гистограмм, здесь возможны 2 варианта описанные выше, это Авто контраст и Выравнивание гистограмм, параметр во второй цепочке генотипа влияет если только в данный момент активен Авто контраст. Во второй и третей ячейке первого генотипа закодированы фильтры, а во втором генотипе их параметры.

**Выбор родителей**

Выбор родителей я осуществлял по такому принципу:

1. Вычислим отношение для каждой особи из потомства

(значение фитнес функции особи / среднее по всему потомству)

1. Целая часть означает сколько раз данная особь войдет в состав родителей.
2. Дроби будут показывать вероятность попадания, если поколение родителей не заполнилось до конца. Просто в цикле будем сравнивать вероятность со случайным значением, пока вероятность не окажется больше, тогда эта особь становить родителем.
3. Так же для того что бы потомство не деградировало лучшая особь сразу переходит в следующее поколение без мутаций. Это может привести к скудности потомства, однако модель которую я предложил довольно простая и поэтому стремиться к локальному максимуму не страшно. Плюс такого подхода заключается в том, что количество шагов в алгоритме заметно уменьшается, за счет того, что алгоритм быстрее сходиться к решению.

**Скрещивание**

Как известно в теории эволюции важную роль играет то, каким образом признаки родителей передаются потомкам. В генетических алгоритмах за передачу признаков родителей потомкам отвечает оператор, который называется скрещивание (его также называют кроссовер или кроссинговер). Этот оператор определяет передачу признаков родителей потомкам. Действует он следующим образом:

1. Из популяции выбираются две особи, которые будут родителями;
2. Определяется (обычно случайным образом) точка разрыва;
3. Потомок определяется как конкатенация части первого и второго родителя.

Так как возможные родители уже выведены в отдельную группу, можно просто случайным образом выбирать двух разных родителей и производить скрещивание. Для скрещивания я использовал обычный одноточечный кроссовер.

**Мутация**

Следующий генетический оператор предназначен для того, чтобы поддерживать разнообразие особей с популяции. Он называется оператором мутации. При использовании данного оператора каждый бит в хромосоме с определенной вероятностью инвертируется. Я использовал 20% инверсию для каждого бита в обоих генотипах.

В принципе для функционирования генетического алгоритма достаточно этих двух генетических операторов, но на практике применяют еще, и некоторые дополнительные операторы или модификации этих двух операторов. Например, кроссовер может быть не одноточечный (как было описано выше), а многоточечный, когда формируется несколько точек разрыва (чаще всего две). Кроме того, в некоторых реализациях алгоритма оператор мутации представляет собой инверсию только одного случайно выбранного бита хромосомы.

**Целевая (фитнес) функция**

Для автоматизации процесса улучшения качества изображения необходимо определить объективный критерий для оценки качества улучшения изображения. Этот критерий используется для по строения целевой функции генетического алгоритма и включает в приведенные выше, оценки изображений. Проблема приведенных функций в том, что они не нормализованы и для каждого изображения значения будут разными, а простоя максимизация параметров фильтров приведет к банальной порче изображений. Для этого мне нужны максимальные значения что бы на них делить(нормализовать). Для этого мне нужно выделить все возможные граничные пиксели. Для этого подойдет подойдёт фильтр резкости с большими значениями (перешарп), после чего получим оценку этого изображения и в силу наблюдений скажем что 50% от этого значения являются целью поиска для генетического алгоритма. Для резкости подойдет та же схема только в этом случаи возьмем 40% от получившейся оценки. Теперь нам надо объединить две оценки. Можно использовать линейную комбинацию, но я решил взять просто их произведение, так как нам важны оба параметра в одинаковой степени, и слишком низкая оценка одного из параметров должна ставить крест на данной особи.

**Схема функционирования генетического алгоритма**

Теперь, перейдем к описанию функционирования генетического алгоритма. Рассмотрим схему функционирования генетического алгоритма в его классическом варианте.

1. Инициировать начальный момент времени t=0t=0. Случайным образом сформировать начальную популяцию, состоящую из k особей. B0={A1,A2,…,Ak}
2. Вычислить приспособленность каждой особи FAi=fit(Ai) i=1…ki=1…k и популяции в целом Ft=fit(Bt) (также иногда называемую термином фитнес). Значение этой функции определяет насколько хорошо подходит особь, описанная данной хромосомой, для решения задачи.
3. Выбрать особь Ac из популяции  Ac=Get(Bt)
4. С определенной вероятностью (вероятностью кроссовера Pc) выбрать вторую особь из популяции Ac1=Get(Bt) и произвести оператор кроссовера Ac=Crossing(Ac,Ac1).
5. С определенной вероятностью (вероятностью мутации Pm) выполнить оператор мутации  Ac=mutation(Ac).
6. С определенной вероятностью (вероятностью инверсии Pi) выполнить оператор инверсии Ac=inversion(Ac).
7. Поместить полученную хромосому в новую популяцию insert(Bt+1,Ac).
8. Выполнить операции, начиная с пункта 3, k раз.
9. Увеличить номер текущей эпохи t=t+1t=t+1.
10. Если выполнилось условие останова, то завершить работу, иначе переход на шаг 2.

**Результаты работы**

Для демонстрации своей работы я написал простую графическую оболочку. Данное приложение может открывать изображения любого типа и любого размера, для этого я воспользовался файлом из библиотеке Шона Барретта - stb\_image.h. Это единственная зависимость в моей программе, весь остальной код включая графическую библиотеку был написан моими руками. Данный алгоритм был проверен в на большом наборе изображений и были выявлены недостатки такого подхода:

1. Цветные изображения сами по себе сложны в редактировании в связи с человеческим восприятием. Некоторые изображения портятся в результате растяжения гистограмм. Это связано с тем что преобразования проходят в цветовом пространстве YCbCr над каналом яркости Y, после обратного преобразования это так же сказывается на цвете пикселей. Поэтому улучшение цветных изображений является отдельной, более сложной задачей.
2. Бывают и такие изображения, которые выглядят лучше при низкой контрастности или резкости. Это изображения с малым количеством деталей, с размытым фоном и другие.
3. Алгоритм проявляет себя лучше всего с дефектными изображениями у которых проблемы с резкостью и контрастностью.

**Время работы и оптимизация реализации**

Для изображения с разрешением 640х425 десять шагов алгоритма выполняются за 0.63 секунды. Для изображения с разрешением 1200х1200 десять шагов выполняются за 3.1 секунды (процессор i7 3770k с тактовой частотой 3.8ГГц). Это итоговые результаты.

Теперь подробнее о том, что я доработал. Для начала я оптимизировал применение операции свертки, а именно случай пикселей на границе рассматривал отдельно, это позволило избавиться от трех if и двух вызовов функции abs в основном цикле. Это уменьшило время применения операции свертки на 50% так как промахов по данным было достаточно много. Так как цикл проходит по всем пикселям изображения это достаточно ощутимая прибавка.

В генетическом алгоритме самая долгая операция – это подсчет значений фитнес функции для каждой особи. Это связано с тем что для начала на надо применить все преобразования к исходному изображению, а затем посчитать его оценку. Оба шага трудоемки, однако процесс получения оценки для каждой особи независим от других поэтому я решил считать каждую особь на отдельном ядре. Так как у меня на процессоре 4 ядра, я сформировал поколение из 4 особой и каждый шаг считал особь на отдельном потоке. Для того что бы не выделять потоки каждую итерацию алгоритма, так как это достаточно затратно, я написал простую реализацию пула потоков и просто добавлял новую задачу в виде подсчета фитнес значения для каждой особи.

Как я сказал выше вся трудоемкость алгоритма заключается в преобразованиях над изображениями, а эта проблема превосходно масштабируется, а это в свою очередь значит, что все эти преобразования можно вычислять на видеокарте, прирост в скорости вычислений может доходить до 100 раз.

**Примеры результатов**

Пример 1.

a - исходное b – результат работы алгоритма

Пример 2.

a - исходное b – результат работы алгоритма

Пример 3.

a - исходное b – результат работы алгоритма

Пример 4.

a - исходное b – результат работы алгоритма

Пример 5.

a - исходное b – результат работы алгоритма