Министерство образования и науки Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»

Институт информационных технологий, математики и механики

**« Преобразования Хафа»**

Выполнил: студент группы 381408

Мальков С. Ю.

Нижний Новгород

2016 г.

**Содержание**

1. Постановка задачи

2. Описание алгоритма

3. Схема распараллеливания

4. Описание программной реализации

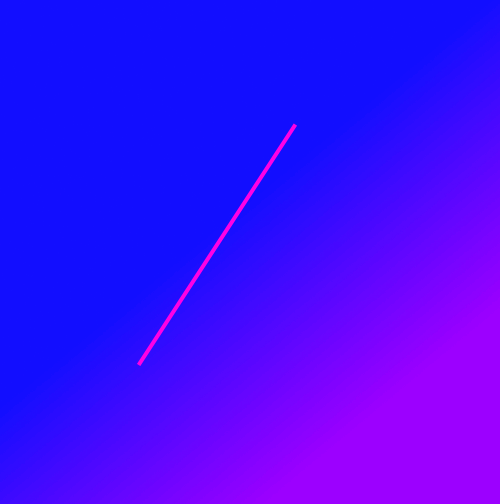
5. Проверка работоспособности

6. Результаты экспериментов

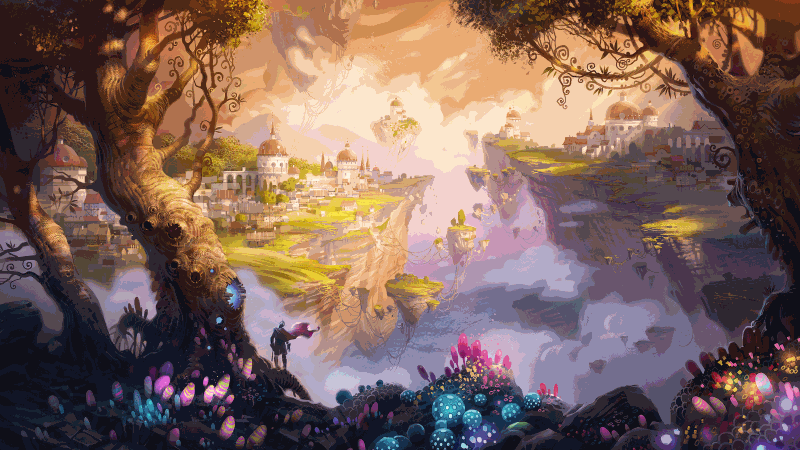
**Постановка задачи**

Реализовать алгоритм для извлечения геометрических элементов из изображения. С помощью MSMPI распараллелить данный алгоритм, добиться ускорения при использовании нескольких процессов, запустить программу на выделенном кластере.

Извлекать геометрический элемент - линию - будем из данного изображения:



Тестировать скорость работы будем на более крупном и детальном изображении:



**Метод решения**

Решать данную задачу будем с использованием преобразований Хафа для прямых.

Преобразование Хафа (Hough Transform) — алгоритм, численный метод, применяемый для извлечения элементов из изображения (патент 1962 г. Поля Хафа). Используется в анализе изображений, цифровой обработке изображений и компьютерном зрении . Предназначен для поиска объектов, принадлежащих определённому классу фигур, с использованием процедуры голосования. Процедура голосования применяется к пространству параметров, из которого и получаются объекты определённого класса фигур по локальному максимуму в так называемом накопительном пространстве (accumulator space), которое строится при вычислении трансформации Хафа.

В основе теории преобразования Хафа лежит утверждение, что любая точка двоичного изображения может быть частью некоторого набора возможных линий:

y = a\*x+b - в декартовых координатах

p = x\*cos(f)+y\*sin(f) - в полярных координатах

Прямую на плоскости можно представить:

x\*cos(f) + y\*sin(f) = R

Где:

R – длина перпендикуляра опущенного на прямую из начала координат,

f — угол между перпендикуляром к прямой и осью OX.

f находится в пределах от 0 до 2\*PI,

R ограничено размерами входного изображения.

Так же каждой точке (R0, f0) пространства (R, f) можно поставить в соответствие счетчик, соответствующий количеству точек (x, y), лежащих на прямой

x\*cos(f0) + y sin(f0) = R0

Представив изображение в виде бинарного изображения контуров мы применим процедуру голосования, затем выберем точку в накопительном пространстве с наибольшим значением и в зависимости от вычисленного угла нарисуем прямую линию.

**Схема распараллеливания**

Мы загрузим изображение, используя библиотеку OpenCV получим контурное изображение, затем передадим контурное изображение всем процессам. У каждого процесса будет свой массив, представляющий фазовое пространство. Процессы будут обрабатывать каждую N строку изображения, где N - номер процесса. После обработки всех строк мы сформируем на каждом процессе свое фазовое пространство. В данном пространстве мы выберем точку и угол с максимальным значением счетчика и сохраним в качестве результата их значения.

Каждый процесс вернет результат в основной процесс, основной процесс в момент получения выберет результат с максимальным значением счетчика, а затем нарисует прямую линию в данной точке под заданным углом.

**Описание программной реализации**

Программа использует библиотеку OpenCV для загрузки изображений, отображения в отдельных окнах, и для создания контурного изображения, а также для освобождения ресурсов в конце программы.

В программе есть параметр SHOW\_WINDOWS, указывающий, надо ли отображать картинку в отдельных окнах.

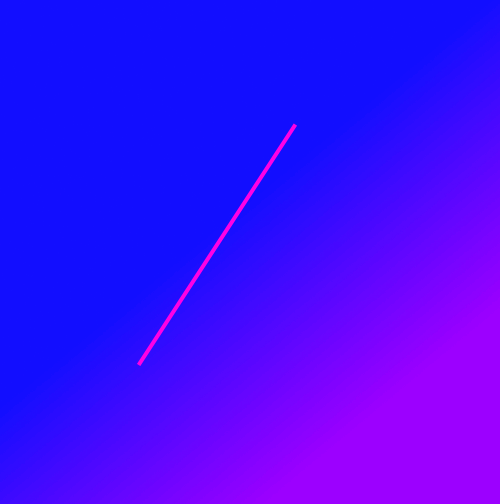
Все операции загрузки, отображения и вывода данных выполняются на главном процессе.

Также есть отдельная реализация программы без использования OpenCV, где сгенерированная контурная картинка загружается из текстового файла, где первые два числа - ширина и высота изображения, а все последующие числа - 1 если белый цвет, 0 если черный цвет. Данная реализация была создана для тестирования на кластере, поскольку он не хочет работать с OpenCV.

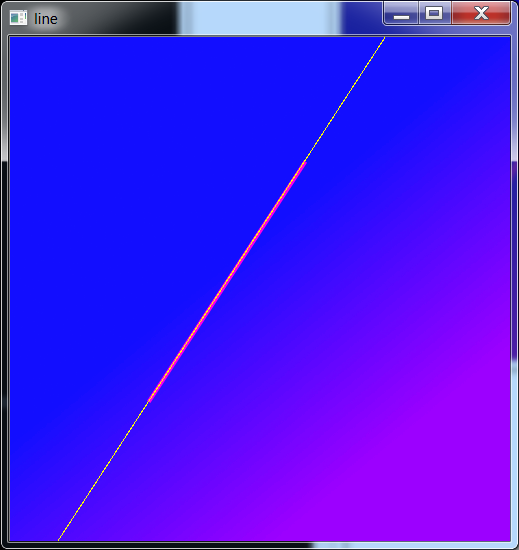
Отсчет времени работы программы начинается после загрузки изображения в память и создания контурного изображения, поскольку нас интересует лишь сам алгоритм преобразований Хафа, заканчивается отсчет времени сразу же после отправки всеми процессами результатов.

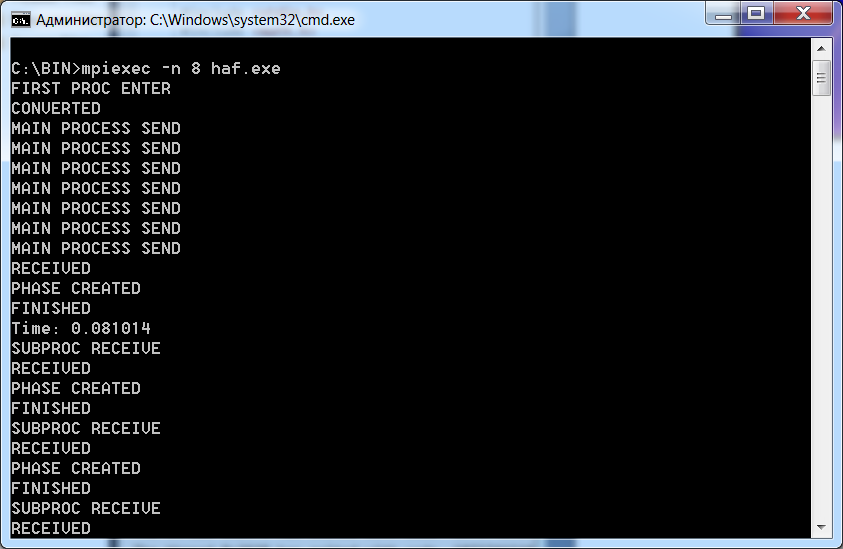
**Подтверждение корректности**

Исходное изображение:



Результат работы программы при восьми процессах:





**Результаты экспериментов**

|  |  |
| --- | --- |
| Количество процессов | Время работы |
| 1 | 24.049759 |
| 2 | 12.105810 |
| 4 | 6.095618 |
| 8 | 3.502482 |
| 16 | 2.214982 |
| 32 | 1.517995 |

Мы видим практически линейный рост производительности в зависимости от увеличения процессов, поскольку основная нагрузка в программе - вложенные циклы, которые мы распараллеливаем.