**Содержание**

[Введение 4](#_Toc134126048)

[1. Разработка и описание послдевательного алгоритма программной системы 6](#_Toc134126049)

[2. Разработка программной системы, реализущей последовательный алгоритм обработки 7](#_Toc134126050)

[3. Разработка и обоснование варианта схемы паралезации алгоритма 8](#_Toc134126051)

[4. Разработка программных модулей системы параллельной обработки данных 9](#_Toc134126052)

[5. Оценка и сравнительный анализ 10](#_Toc134126053)

[Заключение 11](#_Toc134126054)

[Список используемых источников 12](#_Toc134126055)

# Введение

Алгоритм робертса

Он представляет собой первое известное решение задачи об удалении невидимых линий. Алгоритм, прежде всего, удаляет из каждого тела те ребра или грани, которые экранируются самим телом. Затем каждое из видимых ребер каждого тела сравнивается с каждым из оставшихся тел для определения того, какая его часть или части, если таковые есть, экранируются этими телами. Поэтому вычислительная трудоемкость алгоритма Робертса растет теоретически, как квадрат числа объектов. Это в сочетании с ростом интереса к растровым дисплеям, работающим в пространстве изображения, привело к снижению интереса к алгоритму Робертса. Однако математические методы, используемые в этом алгоритме, просты, мощны и точны. Кроме того, этот алгоритм можно использовать для иллюстрации некоторых важных концепций. Работа Алгоритм Робертса проходит в два этапа:

1. Определение нелицевых граней для каждого тела отдельно.

2. Определение и удаление невидимых ребер.

Оператор Робертса используется ради быстроты вычислений, но он проигрывает в сравнении с альтернативами с его значительной проблемой чувствительности к шуму. Он даёт линии тоньше, чем другие методы выделения границ.

MPI - это стандарт на программный инструментарий для обеспечения связи между ветвями параллельного приложения.

MPI расшифровывается как "Message passing interface" ("Взаимодействие через передачу сообщений"). Несколько путает дело тот факт, что этот термин уже применяется по отношению к аппаратной архитектуре ЭВМ. Программный инструментарий MPI реализован в том числе и для ЭВМ с такой архитектурой.

MPI предоставляет программисту единый механизм взаимодействия ветвей внутри параллельного приложения независимо от машинной архитектуры (однопроцессорные / многопроцессорные с общей/раздельной памятью), взаимного расположения ветвей (на одном процессоре / на разных) и API операционной системы.   
( API = "applications programmers interface" = "интерфейс разработчика приложений" )

Программа, использующая MPI, легче отлаживается (сужается простор для совершения стереотипных ошибок параллельного программирования) и быстрее переносится на другие платформы (в идеале, простой перекомпиляцией).

В настоящее время разными коллективами разработчиков написано несколько программных пакетов, удовлетворяющих спецификации MPI, в частности: MPICH, LAM, HPVM и так далее. Они выступают базовыми при переносе MPI на новые архитектуры ЭВМ. Здесь в пособии рассматриваются разновидности MPICH. Это сделано по двум причинам:

* MPICH написан авторами спецификации,
* и наиболее распространен.

Таким образом, далее по тексту термин MPI используется не только для обозначения изложенных в спецификации сведений, но и, в определенной мере, для описания характеристик конкретной базовой реализации.

Минимально в состав MPI входят: библиотека программирования (заголовочные и библиотечные файлы для языков Си, Си++ и Фортран) и загрузчик приложений.

Дополнительно включаются: профилирующий вариант библиотеки (используется на стадии тестирования параллельного приложения для определения оптимальности распараллеливания); загрузчик с графическим и сетевым интерфейсом для X-Windows и проч.

Структура каталогов MPICH выполнена в полном соответствии с традициями Юникса: bin, include, lib, man, src, ... Минимальный набор функций прост в освоении и позволяет быстро написать надежно работающую программу. Использование же всей мощи MPI позволит получить БЫСТРО работающую программу - при сохранении надежности.

# 1. Разработка и описание послдевательного алгоритма программной системы

Цель работы:

Разработать приложение обработки ч/б изображений оператором Робертса с последующим увеличением масштаба

Решение:

1. Преобразовать изображение в битовую маску
2. Обработать изображение оператором Робертса
3. Увеличить масштаба
4. Сохранить результат

Оператор Робертса работает по принципу:

Вычисляет на плоском дискретном изображении сумму квадратов разниц между диагонально смежными [пикселами](https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D0%BA%D1%81%D0%B5%D0%BB%D1%8C). Это может быть выполнено сверткой изображения с двумя ядрами:

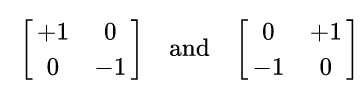


Рисунок 1. Свертка изображения с двумя ядрами

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

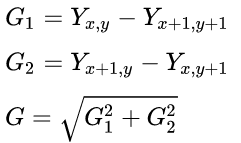


Рисунок 2. Правило вычисления величины перепада

Изменение масштаба будет происходить путем увеличения количества пикселей: каждый пиксель будет вставлен в новую маску раз, где - масштаб.

# 2. Разработка программной системы, реализущей последовательный алгоритм обработки



Рисунок 3. Схема алгоритма последовательной обработки

# 3. Разработка и обоснование варианта схемы параллезации алгоритма

Цель работы:

Разработать мультипроцессорного приложение обработки ч/б изображений оператором Робертса с последующим увеличением масштаба

Решение:

1. Преобразовать изображение в битовую маску
2. Разбиение маски на 4 равных части
3. Обработать каждой части параллельно оператором Робертса
4. Увеличить масштаба каждой части параллельно
5. Собрать результатов
6. Сохранить результат

# 4. Разработка программных модулей системы параллельной обработки данных



Рисунок 4. Схема алгоритма параллельной обработки

# 5. Оценка и сравнительный анализ

Пример обработки изображения:



Рисунок 5. Исходное изображение



Рисунок 5. Результат

# Заключение

Реализовал последовательную и параллельную программу обработки изображения. Сравнил результаты работы последовательно и параллельной программы. Показал примеры обработки изображения

# Список используемых источников

* <https://learn.microsoft.com/ru-ru/dotnet/api/system.drawing.bitmap?view=dotnet-plat-ext-7.0>
* https://github.com/mpidotnet/MPI.NET