Библиотека IPLab и ее временная оптимизация

И.В.Богин, Е.В. Чепин
Московский Инженерно-Физический Институт (Государственный университет)
Москва, Россия

Abstract

This article describes new features of last version of image processing library IPLab v. 2.1. There is short review of the most popular universal image-processing library in article. Article also contains description of library functions and productivity optimization. The optimization has based on algorithms, process and memory peculiarity. The fastest compiler for those set of function has been chosen. The result of optimization describes in detail in main part of the article.

Keywords: Обработка, изображения, оптимизация, производительность, компилятор, IPLab

1. ВВЕДЕНИЕ

На данный момент отрасль программной и аппаратной обработки изображений занимает очень значимую роль в области высоких технологий. Сфера применения программных и аппаратных средств обработки изображений простирается от бездонных океанских расщелин до необъятных просторов космоса. Такой размах обусловлен многообразием практических задач, использующих либо сами изображения, либо результаты их анализа. Среди них, и автономное управление движущимися объектами, и визуализация изображений в недоступных человеческому глазу диапазонах, и количественная оценка параметров объектов, и практически необъятный раздел распознавания образов, пересекающийся с направлением создания искусственного интеллекта. Естественно, что сложные процедуры обработки изображения, в основном, базируются, используют или требуют предварительного вызова более простых процедур. В связи с чем, возникла необходимость в универсальных программных библиотеках, предоставляющих базовый набор функций обработки изображения.

2. ОБЗОР УНИВЕРСАЛЬНЫХ ПРОГРАММНЫХ БИБЛИОТЕК ОБРАБОТКИ ИЗОБРАЖЕНИЙ

На данный момент на рынке представлено достаточно большое количество универсальных программных библиотек. Сведения о них суммированы в таблице 1. Из анализа этой таблицы можно сделать выводы, что базовый набор функций, включающий в себя фильтрацию изображения, геометрические преобразования, поточечные, логические и арифметические преобразования, у большинства библиотек очень похож и предоставляется в большинстве случаев бесплатно. Но как только функциональность библиотеки выходит за рамки этого базового набора , ее стоимость резко увеличивается до уровня нескольких тысяч долларов. Также стоит отметить факт непредоставления фирмами исходного кода библиотек. Причины этого вполне очевидны, но пользователи оказываются лишены возможности переносить эти библиотеки на другие аппаратные или программные платформы. В связи с этим наличие текстов библиотеки IPLab, позволяет адаптировать библиотеку под конкретные нужды пользователя.

Таблица 1. Сравнение универсальных библиотек обработки изображения

Название (Фирма)	Цена	Доступность исходных кодов	Число функций
IPL (Intel)	Бесплатно	Нет	Около 200
MIL(Matrox)	4000$	Нет	Около 400
Sapera 4.1 (Coreco)	3900$	Нет	Около 300
eVision (Euresys)	Модули можно приобретать отдельно	Нет	
Impact (Matrix Vision)	Базовый бесплатен.	Нет	
Halcon 6.1 (Mvtec)	4600 Euro	Нет	990
IPLab (Mephy System)		Да	130

Таблица 1. Краткие сведения об универсальных библиотеках обработки изображения.

3. НОВАЯ ВЕРСИЯ БИБЛИОТЕКИ IPLAB И ЕЕ ХАРАКТЕРИСТИКИ

Первоначально библиотека была создана в среде разработки Borland C++ 5.02(в дальнейшем BC5) и к ней прилагалось тестовое приложение для запуска и измерения производительности функций, также созданное в этом пакете[1,2]. В связи с невысокой производительностью библиотеки, существенно отставшей от производительности библиотек других фирм, было принято решение о переводе библиотеки в другую среду разработки.

В качестве новой среды разработки была избранна Visual Studio 6.0, в качестве компиляторов использовался родной для этой среды компилятор Visual C++ 6.0 (в дальнейшем VC6) и удобно интегрирующийся с ней компилятор Intel C++ 5.0 (в дальнейшем IC5). Выбор именно этих компиляторов был обусловлен их высокой производительностью [3-5]. Как показали дальнейшие измерения, один этот шаг позволил в 2 раза увеличить производительность. После того как библиотека и тестовое приложение были перенесены в среду разработки Visual Studio 6.0, была проведена отладка функций библиотеки и тестового приложения.

На данный момент библиотека IPLab MSCo включает в себя свыше 130 функций различного назначения, условно их можно классифицировать следующим образом:

- Арифметические и логические функции

  Основные логические(And,Or,Xor,Not) и арифметические (+,-,*,/) операции, выполняющиеся над каждым пикселем изображения, где первым операндом служит значение пикселя, а вторым заданное пользователем число.

- Функции совместной обработки двух изображений

  Логические(And, Or, Xor), арифметические (+,-) функции, а также функции сравнения (равно, не равно, больше, больше равно и т. д.), где операндами служат соответствующие пиксели двух изображений.

- Функции бинаризации изображения

  Перевод изображения в бинарное из двух заданных цветов, где значение пикселя переводится в цвет 1, если он лежит выше порогового значения (ниже, между двух пороговых значений, вне пороговых значений), а в обратном случае присвоение ему цвета 2.

- Общая фильтрация изображения

  Медианный фильтр, фильтры поиска максимума и минимума, линейные высокочастотные и низкочастотные фильтры.

- Простейшие функции редактирования изображения

  Рисование прямых линий заданного цвета и размера, установка строк (столбцов или их частей) в максимальное (минимальное) встреченное значение в данной строке (столбце или их части).

- Простейшие статистические функции

  Подсчет пикселей заданного цвета в строке (столбце). Нахождение минимального (максимального) значения пикселя в данной строке (столбце, всем изображении).

- Геометрические преобразования

  Повороты на заданные(90 ,180 ,270 градусов) или произвольные углы, зеркальные отражения относительно горизонтальных и вертикальных осей, операции масштабирования и аффинного преобразования, сдвиг (циклический и не циклический) изображения в заданную точку.

- Функции выделения контуров

  Курсовые градиентные маски, фильтры Превита, Лапласа, Робертса, Собеля, Кирша, в различных вариациях (различное количество одновременно обслуживаемых масок, различные стратегии подсчета результата свертки и т.д.)

- Функции выделения линий

  Фильтры выделения горизонтальных вертикальных, диагональных линий.

- Морфологические функции

  Функции “расширения”, “разъедания”, “открытия” и “закрытия“ изображения.

- Функции для работы с контурами

  Различные функции утончения контуров, а также различные варианты алгоритма ”жука” для получения массива координат контуров.

- Функции для работы с гистограммой изображения

  Функции изменения яркости, контрастности, общей гистограммы и ее частей.

- Специальные функции

  Функции для осуществления преобразования Хоха, выделения контуров без использования фильтров, заполнения контуров. Функции для распознавания образов на основе атрибутных грамматик.

Более подробное описание функций можно найти в справочной системе тестовой программы IPLab или в соответствующей литературе[8-11].

Объем динамически загружаемой библиотеки порядка 150 килобайт(iplab.dll). Тестовое приложение (IPLab.exe) занимает порядка 150 килобайт, с его помощью можно загрузить изображение в формате BMP и вызвать для его обработки любую из функций библиотеки. Из тестового приложения можно вызвать справочную систему формата WinHelp содержащую подробную информацию о механизмах и особенностях работы каждой из функций.

4. ТАЙМИРОВАНИЕ И ВРЕМЕННАЯ ОПТИМИЗАЦИЯ БИБЛИОТЕКИ IPLAB

Было проведено таймирование функций из тестового приложения для библиотек скомпилированных с помощью различных компиляторов (Borland C++ 5.0, Visual C++6.0, Intel C++ 5.0). Данный тест проводился на компьютере Celeron 333, 64 Mb RAM 4Mb VRAM Win98. В качестве тестового изображения использовался файл 672x473x256 GrayScale.. Для всех компиляторов тестировалась Release версия с Maximize Speed оптимизацией.. Каждая функция была запущенна 30 раз и в качестве финального результата вычислялось среднее арифметическое. Результаты показали уверенное лидирование компилятора IC5 (его преимущество на фильтрах достигает 1.5 –2 раз по сравнению с VC6), и сильное отставание компилятора BC5 практически для всех функций.

После выбора компилятора обеспечивающего наибольшую производительность в нашем случае им оказался Intel C++ 5 ,было проведено сравнение с библиотекой Image Processing Library от фирмы Intel (в дальнейшем IPL). Сравнение производительности нашей библиотеки с IPL выявило существенное отставание нашей библиотеки на таких

категориях функций, как общая фильтрация, выделение границ, геометрические преобразования (использующие аффинное преобразование), морфологические операции.. Возникла необходимость оптимизации производительности нашей библиотеки. Функции, на которых она сильно отставала от IPL можно разделить на две категории:

а) Использующие алгоритм свертки.

б) Использующие другие механизмы просмотра.

Среди сильно отстающих от IPL первых было явное большинство и поэтому оптимизацию было решено начать с них, так как после нахождения общего механизма оптимизации его можно было бы применить сразу для всех функций из данной категории с незначительными изменениями для каждой из подгрупп функций. В качестве тестовых функций для оптимизации были выбраны следующие три фильтра:

- Previtt (обрабатывает 4 маски одновременно) производительность в 37 раз хуже (Borland C++ 5.02) , чем у IPL (обрабатывает 1 маску одновременно). Для IPL вызывалась функция FixedFilter с параметром IPL_PREWITT_3x3_V.
- Laplas1 (обрабатывает 2 маски одновременно) в 28 раз хуже (Borland C++ 5.02), чем у IPL(обрабатывает 1 маску одновременно). Для IPL вызывалась функция FixedFilter с параметром IPL_LAPLASIAN_3x3.
- Laplas2 (обрабатывает 1 маску одновременно) в 24 раза хуже (Borland C++ 5.02), чем у IPL (обрабатывает 1 маску одновременно). Для IPL вызывалась функция FixedFilter с параметром IPL_LAPLASIAN_3x3.

Можно указать следующие возможные пути оптимизации текста программ:

1)Использование инструкций MMX

2)Ускорение масштабирования результата

3)Исключение влияния выравнивания изображения по границе 4(16) байт.

4)Сохранение частичных сумм свертки

4.1 Использование инструкций MMX

В качестве первого шага оптимизации было решено переписать на ассемблере, используя инструкции MMX, внутренний цикл свертки для каждого положения окна просмотра. Так как MMX поддерживает формат 64 разрядного слова, а результат свертки укладывается в 16 разрядов, появилась возможность одновременно обработать 4 положения апертуры вместо одного, то есть теоретический выигрыш должен был быть в 4 раза, но после этого выполняется масштабирование результата свертки, чтобы его размер помещался в байт. На данный момент библиотека поддерживает 8-битный , 24-битный и 32-битный форматы изображения. В случае 8-битного изображения значение каждого из пикселей умещается в 1 байт, в случае 24-битного – три отдельно обрабатываемые цветовые составляющие RGB занимают по 1-му байту, аналогично в 32-битном изображении, с той лишь разницей что там еще добавляется 8-битный альфа –канал. То есть масштабирование результата свертки до размеров одного байта необходимо для всех поддерживающихся типов изображения. Масштабирование использует операцию деления, что «съедает» большую часть выигрыша достигнутого использованием MMX для оптимизации внутреннего цикла. Вполне естественно, что следующим пунктом общей оптимизации стала операция масштабирования, но сначала о результатах полученных в этом пункте.

Прирост производительности составил приблизительно одну треть на функциях Previtt (обрабатываютя 4 маски одновременно), Laplas1 (2 маски одновременно) и Laplas 2 (1 маска). В общем, использование MMX и компилятора Intel C++ 5.0 дает двукратный выигрыш в скорости по сравнению с компилятором Visual C++ 6.0, без использования MMX.

4.2 Ускорение масштабирования результата

Проведение масштабирования необходимо, так как часто результат свертки не «влезает» в байт, а если просто отбрасывать старший байт результата свертки, то изменение цветов будет непропорциональным, что плохо скажется на качестве результирующего изображения). Вместо деления на коэффициент масштабирования koef = kmax \ 255, где kmax – максимальное значение результата свертки, поделим на 2^k , k вбирается таким образом, чтобы 2^(k-1) < koef < 2^k , Это не изменит правильности масштабирования, так как его цель пропорционально уменьшить значения цвета всех пикселей, чтобы они помещались в байт, так что, если мы несколько превысим коэффициент масштабирования это отразится лишь на уменьшение контрастности, но не на пропорциональности цветов. Также цикл масштабирования был переписан, используя инструкции MMX, обрабатывая одновременно по 4 значения свертки за 1 команду.

Прирост производительности составил 200% на функции Previtt и 300% на функциях Laplasian 1 Max и Laplasian 2 (имеется в виду по сравнению с вариантом, где оптимизации подвергался лишь внутренний цикл свертки). Следует отметить, что здесь уже разницы между IC5 и VC6 нет вообще, что говорит о том, что до этого большую часть выигрыша IC5 составляла оптимизация вычислений с плавающей запятой. Общий выигрыш составил по сравнению с BC5 – 500% (previtt) и 600% (laplas) , с VC6 – 400%(previtt) и 500% (laplas) , с IC5 – 200% (previtt) и 300% (laplas).

4.3 Исключение влияния выравнивания изображения по границе 4(16) байт

Проверялось влияние выравнивания изображения по границе 4(16) байт. Это достигалось путем отбрасывания первых нескольких пикселей, до границы кратной 4(16-байтам).

Никакого существенного влияния не обнаружено, (хотя это должно было увеличить скорость копирования в кеш[7]). Видимо компилятор в большинстве случаев самостоятельно выравнивал массив изображения в памяти

4.4 Сохранение частичных сумм свертки

Было принято решение запоминать частичные суммы свертки, для ускорения подсчетов. Большинство фильтров обладают симметричными первой и третьей строкой. Вычислив свертку по третьей строке одного положения аппретуры (с центром в точке X,Y) и запомнив эту частичную сумму, для положения апертуры с центром находящимся в точке X,Y+2, т.е. на две строчки ниже, можно не вычислять свертку по первой строке, а воспользоваться запомненной сверткой по третьей строке первого фильтра.

Практически не удалось добиться ускорения (всего лишь на 1 мс). Исходный единый цикл, пришлось разбить, выделив отдельно обработку первых двух строк, где частичные суммы только запоминаются, но не используются, а оставшуюся часть представить в виде вложенного цикла, где внутренний цикл пробегает по двум строчкам используя запомненные частичные суммы, и записывая на их место новые. И внешний цикл, где восстанавливается указатель на начало массива частичных сумм, после его полного цикла чтения и перезаписи. Видимо из-за дополнительных расходов по организации цикла и большим числом копирований в память, эта методика не принесла предполагаемого увеличения скорости.

Результаты оптимизации функции свертки отображены на Рис. 1. Под номером 1 изображается время выполнения функции до оптимизации. Под номером 2 оптимизация цикла свертки за счет применения инструкций MMX. Под номером 3 помимо цикла свертки оптимизирован и цикл масштабирования. Под номером 3 к методикам, используемым в пункте 2, добавляется выравнивание и сохранение частичных сумм свертки.

В докладе приводятся результаты таймирования и оптимизации большого количества функций библиотеки IPLab.

5. ЗАКЛЮЧЕНИЕ

Основной результат повышения производительности дало использование инструкций MMX и упрощение операций повторяющихся для каждой точки. В целом библиотека IPLab несколько отстает по производительности от других универсальных библиотек, но наличие ее исходных кодов дает возможность гибкой адаптации к различным платформам, а также оставляет возможность дальнейшего увеличения функциональности и производительности. Одним

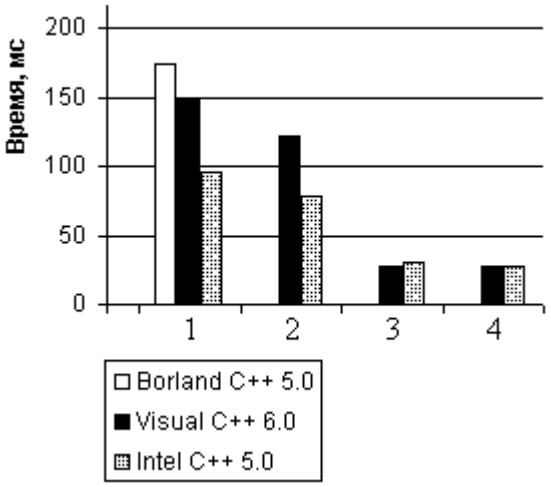

Рис. 1. **Результаты оптимизации функции свертки**.

из возможных путей ускорения является использование предвыборки в кеш данных, которые должны подвергнуться обработке в ближайшее время. На всех современных процессорах (Atlon, Pentium III, Pentium IV) существует такое семейство инструкций, но недостаток его состоит в том, что для каждого процессора тактика предвыборки должна быть различна, к тому же инструкции семейств процессоров Intel и AMD несовместимы между собой.

Принципиально другая возможность ускорения времени выполнения алгоритмов обработки изображений состоит в использовании мультипроцессорных ВС (МПВС), например [12,13]. Однако представляется разумным перед разработкой и проведением временной оптимизации параллельных алгоритмов для мультипроцессорной системы той или иной архитектуры иметь в наличии оптимизированные программные коды на однопроцессорной системе. Причем, необходимо четко представлять, за счет каких «механизмов» и технологий получено ускорение, поскольку при переходе на МПВС

6. ЛИТЕРАТУРА

[1]И.В Бодров, П.Н.Попов, Е.В.Чепин. Разработка библиотеки функций для обработки изображений.- Тез. II межд. н.-т. конф. "Новые информационные технологии и системы", г.Пенза, 1996.

[2]И.В.Бодров, А.Ю.Каталов, Е.В.Чепин. Разработка библиотеки функций для обработки изображений (Designing of Functions Library for Image Processing) -Научная сессия МИФИ-98., Сб.научн. трудов. В13 томах. Т.7., М.: МИФИ, 1998, с.194-197.

[3]Intel® Image Processing Library v. 2.5 Documentation.

[4]К. Касперски. Сравнительный анализ оптимизирующих компиляторов C/C++/ “Программист”, № 4, 2002г., с.18 - 27, “Викфилд”.

[5]В.Чистяков, “Кто сегодня самый шустрый?”, ж. «Технология Клиент - Сервер” 3-ий квартал 2001 г., “Оптим.Ру”.

[6]В.Чистяков, “Кто сегодня самый шустрый –? 2”, ж. «Технология Клиент - Сервер” 4-ый квартал 2001 г., “Оптим.Ру”.

[7]К. Касперски. Секреты копирования памяти./ “Программист”, № 3, 2002, с.10 - 17, “Викфилд”.

[8]У.Прэтт. Цифровая обработка изображений - В 2-х кн. М.:Мир, 1982 г.

[9]"Быстрые алгоритмы в цифровой обработке изображений. Преобразования и медианные фильтры" Под ред. Т.С.Хуанга, Москва, "Радио и связь", 1984 г.

[10]В.Яншин, Г.Калинин, "Обработка изображений на языке Си для IBM PC алгоритмы и программы", Москва, "Мир", 1994 г.

[11]Р.Е.Быков, С.Б. Гуревич. Анализ и обработка цветных и объемных изображений, Москва, "Радио и связь", 1984 г.

[12]A.V.Arshavsky, E.V.Chepin, O.A.Levchenko, A.N.Nikitin. Time Parameters of Some Parallel Image Processing Algorithms of Hardware Complex – TMS320C40+NM6403. Proc. of the 2nd Int. Workshop on Comp. Science and Inf. Techn.- CSIT'2000, Ufa, Russia, 2000, vol.2, p.29-37.

[13]А.В.Кириллов, К.Е.Лепешенков, Е.В.Чепин. Использование возможностей многопроцессорных систем в пакете "РЕНТГЕН" (JEK-F v.8.2) Научная сессия МИФИ-2002. Сб.научн. трудов. В14 томах. Т.12., М.: МИФИ, 2001, с.188.

Об авторах

И.В.Богин, студент Московского Инжинерно-Физического Института(Государственного университета).

E-mail: **ilya_bogin@rambler.ru**

Е.В. Чепин, к.т.н., с.н.с., доцент кафедры «Компьютерные системы и технологии» Московского Инженерно-Физического Института (Государственного университета).

E-mail: **chepin@dozen.mephi.ru**